1-1-2013

Discovery of communications patterns by the use of intelligent reasoning

J Fulcher
University of Wollongong, john@uow.edu.au

Minjie Zhang
University of Wollongong, minjie@uow.edu.au

Q Bai
University of Wollongong, quan@uow.edu.au

F Ren
University of Wollongong, fren@uow.edu.au

Discovery of communications patterns by the use of intelligent reasoning

Abstract

An agent-based model of communications traffic generated within a social network is described, which caters for various knowledge discovery techniques to be used in order to extract 'interesting' temporal patterns contained within anomalous data records. Attendant Java-based software — NetShow — is presented which enables analysis and display of static network configuration, links between network nodes, and correlations between the traffic passing through nominated nodes. In addition, display of network dynamics is facilitated via the incorporation of swarm techniques. Related issues of 'similarity', 'familiarity' and 'contact lists' are discussed within the context of Social Network Analysis and Link Mining. Finally, several suggestions are made for extending the work reported earlier in the chapter.

Keywords

patterns, intelligent, communications, reasoning, discovery

Disciplines

Engineering | Science and Technology Studies

Publication Details

Fulcher, J., Zhang, M., Bai, Q. & Ren, F. (2013). Discovery of communications patterns by the use of intelligent reasoning. In K. Nakamatsu & L. Jain (Eds.), Handbook on Reasoning-Based Intelligent Systems (pp. 433-462). Singapore: World Science.

Chapter 16

DISCOVERY OF COMMUNICATIONS PATTERNS BY THE USE OF INTELLIGENT REASONING

J. Fulcher, M. Zhang, Q. Bai and F. Ren
Intelligent Systems Research Centre,
University of Wollongong NSW 2522 Australia
{*john,minjie,quan,fr510*}*@uow.edu.au*

An agent-based model of communications traffic generated within a social network is described, which caters for various knowledge discovery techniques to be used in order to extract 'interesting' temporal patterns contained within anomalous data records. Attendant Java-based software — `NetShow` — is presented which enables analysis and display of static network configuration, links between network nodes, and correlations between the traffic passing through nominated nodes. In addition, display of network dynamics is facilitated via the incorporation of swarm techniques. Related issues of 'similarity', 'familiarity' and 'contact lists' are discussed within the context of Social Network Analysis and Link Mining. Finally, several suggestions are made for extending the work reported earlier in the chapter.

16.1. Data Ming and Knowledge Discovery in Databases

Data is accumulating at TerraByte (PetaByte even) rates on a weekly, daily, or even hourly basis, depending on the application domain of interest — in any case, far beyond our ability to keep pace with extracting meaningful information from the relevant data bases and/or data warehouses in question. Given this phenomenon (which, by the way, shows no signs of abating, but rather will continue to increase in accordance with Moore's Law), the huge up swell of interest in Data Mining (DM) during the past two decades should come as no surprise. DM combines techniques developed separately within the fields of DataBases, Machine Learning (ML) and Statistics, and especially those capable of handling large data sets.[1,2]

The fundamental aim with DM is to discover 'patterns of interest' hidden within the data. Just what constitutes 'interesting' is nevertheles open to interpretation. More specifically, various stakeholders can hold quite different views in this regard — for instance a Data Miner may not have the necessary background/experience/insight to interpret such patterns. Furthermore, the 'customer'

will often not themselves know what specific patterns to look for — in which case the DM exercise is essentially one of exploration.

There are a number of critical considerations in DM, most fundamentally the nature of 'information' itself. First and foremost, data and information are *not* synonymous. According to Fulcher, information can be defined as 'data + meaning' (and by extension, knowledge as 'information + understanding'; wisdom as 'knowledge + experience + insight', and so forth).[3] In other words, data of itself is of little value; the 1s and 0s representing the data stored within the data bases/warehouses of interest need to be interpreted *in context* before they will be of value to stakeholder(s). An analogy can be drawn here with internet search engines. The common fallacy underlying the latter can be paraphrased as follows: "there is so much information out there on the net, all one has to do is go searching". Not so — in fact the World Wide Web (WWW) can be characterized as exhibiting a very small signal-to-noise ratio — SNR (to use signal processing terminology). The adage "not to believe everything one reads" in the print media needs to be applied even more stringently when viewing outputs produced by internet search engines. Indeed, one needs to exercise considerable caution in assigning credibility and authority as a matter of course (i.e. by default), devoid of any requisite critical evaluation.

A second fundamental issue with DM is data pre-processing, for indeed DM is only one step (albeit it the most central one) of the broader Knowledge Discovery in Databases (KDD) process, as outlined in Table 16.1.

Prior to DM, we need to undertake data pre-processing, else we run the risk of encountering 'garbage in, garbage out'. Ideally we would like to be working with a 'knowledge rich' data set, but more typically this will not be the case; there will be missing, redundant, inaccurate, inconsistent and/or noisy data in practice. Error bounds checking, noise filtering, data normalization, and format conversion are some typical data 'cleaning' steps that need to be undertaken prior to DM proper.

Another consideration is the use of simple data visualization tools prior to invoking DM techniques in earnest. For example, data clustering can often become immediately apparent by way of 2D(3D) visualization; likewise scatter plots can reveal dependencies within the data set.

Turning now to the central step of the KDD process, we have at our disposal several DM techniques from which we can choose — Association Rules (AR), Decision Trees (DT) and Artificial Neural Networks (ANN) being three of the more popular. The former originated from the field of Databases, the next from the Machine Learning (ML) and Statistics fields, and the latter from the connectionist branch

Table 16.1. KDD Process.

1	Data selection (extraction)
2	Data pre-processing (cleaning, enrichment, coding, etc.)
3	Data Mining proper
4	Interpretation of discovered patterns
5	Presentation/reporting of results

Chapter 16

DISCOVERY OF COMMUNICATIONS PATTERNS BY THE USE OF INTELLIGENT REASONING

J. Fulcher, M. Zhang, Q. Bai and F. Ren
Intelligent Systems Research Centre,
University of Wollongong NSW 2522 Australia
{*john,minjie,quan,fr510*}*@uow.edu.au*

An agent-based model of communications traffic generated within a social network is described, which caters for various knowledge discovery techniques to be used in order to extract 'interesting' temporal patterns contained within anomalous data records. Attendant Java-based software — `NetShow` — is presented which enables analysis and display of static network configuration, links between network nodes, and correlations between the traffic passing through nominated nodes. In addition, display of network dynamics is facilitated via the incorporation of swarm techniques. Related issues of 'similarity', 'familiarity' and 'contact lists' are discussed within the context of Social Network Analysis and Link Mining. Finally, several suggestions are made for extending the work reported earlier in the chapter.

16.1. Data Ming and Knowledge Discovery in Databases

Data is accumulating at TerraByte (PetaByte even) rates on a weekly, daily, or even hourly basis, depending on the application domain of interest — in any case, far beyond our ability to keep pace with extracting meaningful information from the relevant data bases and/or data warehouses in question. Given this phenomenon (which, by the way, shows no signs of abating, but rather will continue to increase in accordance with Moore's Law), the huge up swell of interest in Data Mining (DM) during the past two decades should come as no surprise. DM combines techniques developed separately within the fields of DataBases, Machine Learning (ML) and Statistics, and especially those capable of handling large data sets.[1,2]

The fundamental aim with DM is to discover 'patterns of interest' hidden within the data. Just what constitutes 'interesting' is nevertheles open to interpretation. More specifically, various stakeholders can hold quite different views in this regard — for instance a Data Miner may not have the necessary background/experience/insight to interpret such patterns. Furthermore, the 'customer'

will often not themselves know what specific patterns to look for — in which case the DM exercise is essentially one of exploration.

There are a number of critical considerations in DM, most fundamentally the nature of 'information' itself. First and foremost, data and information are *not* synonymous. According to Fulcher, information can be defined as 'data + meaning' (and by extension, knowledge as 'information + understanding'; wisdom as 'knowledge + experience + insight', and so forth).[3] In other words, data of itself is of little value; the 1s and 0s representing the data stored within the data bases/warehouses of interest need to be interpreted *in context* before they will be of value to stakeholder(s). An analogy can be drawn here with internet search engines. The common fallacy underlying the latter can be paraphrased as follows: "there is so much information out there on the net, all one has to do is go searching". Not so — in fact the World Wide Web (WWW) can be characterized as exhibiting a very small signal-to-noise ratio — SNR (to use signal processing terminology). The adage "not to believe everything one reads" in the print media needs to be applied even more stringently when viewing outputs produced by internet search engines. Indeed, one needs to exercise considerable caution in assigning credibility and authority as a matter of course (i.e. by default), devoid of any requisite critical evaluation.

A second fundamental issue with DM is data pre-processing, for indeed DM is only one step (albeit it the most central one) of the broader Knowledge Discovery in Databases (KDD) process, as outlined in Table 16.1.

Prior to DM, we need to undertake data pre-processing, else we run the risk of encountering 'garbage in, garbage out'. Ideally we would like to be working with a 'knowledge rich' data set, but more typically this will not be the case; there will be missing, redundant, inaccurate, inconsistent and/or noisy data in practice. Error bounds checking, noise filtering, data normalization, and format conversion are some typical data 'cleaning' steps that need to be undertaken prior to DM proper.

Another consideration is the use of simple data visualization tools prior to invoking DM techniques in earnest. For example, data clustering can often become immediately apparent by way of 2D(3D) visualization; likewise scatter plots can reveal dependencies within the data set.

Turning now to the central step of the KDD process, we have at our disposal several DM techniques from which we can choose — Association Rules (AR), Decision Trees (DT) and Artificial Neural Networks (ANN) being three of the more popular. The former originated from the field of Databases, the next from the Machine Learning (ML) and Statistics fields, and the latter from the connectionist branch

Table 16.1. KDD Process.

1	Data selection (extraction)
2	Data pre-processing (cleaning, enrichment, coding, etc.)
3	Data Mining proper
4	Interpretation of discovered patterns
5	Presentation/reporting of results

of Artificial Intelligence — AI (and which in more recent times, constitutes one of the three central pillars of Computational Intelligence — CI[4]).

Up to now we have assumed *static* patterns within the data set of interest. Quite a lot of data captured in the real world is however inherently temporal in nature. Not surprisingly, there have been quite a number of techniques developed that facilitate the discovery of *dynamic* (that is, time-varying) patterns. In statistics, the Auto-Regressive Moving Average (ARMA) and Box-Jenkins models have proved popular, while common CI approaches include software agents,[5–7] Time-Delay Neural Networks (TDNNs),[8] Higher-Order Neural Networks (HONNs),[9] Support Vector Machines (SVMs),[10,11] Evolutionary Algorithms (EAs),[12–18] and Genetic Programming (GP).[19–22]

16.1.1. *Communications Data*

The first data set of interest in the present study is communications traffic — a total of 50,000 records — over a 2-year period. The data format is as shown in Table 16.2.

Data always exists for the first two columns in the record, but not necessarily with the third. Uni-directional data is the norm, although some bi-directional data is also contained within this data set. The anonymous data reflects communications activity over a single (tightly interconnected) network during the specified time period. Note that this does not necessarily correspond to people *per se*, but communications services. This network comprises a tightly-connected 'inner circle', with branches to a peripheral group, although some communications do emanate from outside this inner core. In other words, the data describes a social network of sorts.[23] Accordingly, in the next section we turn our attention to Social Network Analysis (SNA).

A second, more extensive (258,052 record), 2-year communications traffic data set was also made available during the course of this study (Table 16.3); it differs from the first data set in that the type of communication is also recorded.

16.2. Social Network Analysis

At its most fundamental level, Social Network Analysis (SNA) models 'entities' (such as people), and the relationships between them, as nodes and links

Table 16.2. Data Format #1.

ID	Date	Time & duration

Table 16.3. Data Format #2.

(sanitized) date & time	'from' service	'to' service

(respectively) on a graphical model.[24–26a] Such graphs can be likened to the entity-relationship (ER) diagrams commonly encountered in databases.[27–29]

While SNA emphasizes network structure, it also encompasses concepts such as 'between-ness', 'closeness', 'centrality' (placement of the most important/significant entities in the middle of graphs), 'clustering coefficient', 'cohesion', 'path length', 'radiality', 'reach', and the like.

Commonly encountered SNA software tools[30b] include `UCINet`,[c] `Pajek`,[31d] and the *network* software[e] developed for the `R` statistical analysis package.

SNA has been successfully applied to criminal investigations.[32,33]

16.3. Intelligent Reasoning Methods

Numerous approaches have been used for temporal data mining, including graph-based methods,[34,35] fractals,[36] Hidden Markov Models (HMMs),[37,38] Bayesian (Belief) networks,[37,39–41] transactional networks,[42] matrix decompositions,[43] and even Semantic Web-related techniques (e.g. RDF, RDFS and OWL).[44] In the present study we focus on the following techniques: (i) link mining, (ii) software agents, (iii) swarms, and (iv) Artificial Neural Networks.

16.3.1. *Link Mining*

Link (or relational) Mining is a specialized form of DM better suited to the mining of highly structured, heterogeneous data sets in which the objects (entities) are linked in some manner.[45] In graphical terms, objects (entities) correspond to nodes, whereas links (relationships) are represented by edges. 'Similarity' depends not only on comparing attributes but also on links (relationships) between objects (entities) — issues of familiarity and similarity are discussed further in Sects. 16.5.4 and 16.9, respectively. Link Mining has been driven by developments in fields as diverse as link analysis, graph mining, relational learning, inductive logic programming, but most especially hypertext and web mining. Link analysis is used in law enforcement, counter-intelligence, fraud detection and the like.[46] Fawcett and Provost investigated cellular (mobile) phone fraud.[47,48] It has the capability of *automatically* constructing link diagrams from large databases. A typical link analysis tool is `NetMap`.[49]

Link Mining has been applied to web page rankings produced by internet search engines, epidemiology (the spread of communicable diseases), and bibliography searching.

[a] http://www.orgnet.com/sna.html
[b] http://www.insna.org/INSNA/soft_inf.html
[c] http://www.analytictech.com/ucinet/ucinet.htm
[d] http://vlado.fmf.uni-lj.si/sub/networks/pajek/
[e] http://cran.r-project.org/src/contrtib/Descriptions/network.html

of Artificial Intelligence — AI (and which in more recent times, constitutes one of the three central pillars of Computational Intelligence — CI[4]).

Up to now we have assumed *static* patterns within the data set of interest. Quite a lot of data captured in the real world is however inherently temporal in nature. Not surprisingly, there have been quite a number of techniques developed that facilitate the discovery of *dynamic* (that is, time-varying) patterns. In statistics, the Auto-Regressive Moving Average (ARMA) and Box-Jenkins models have proved popular, while common CI approaches include software agents,[5–7] Time-Delay Neural Networks (TDNNs),[8] Higher-Order Neural Networks (HONNs),[9] Support Vector Machines (SVMs),[10,11] Evolutionary Algorithms (EAs),[12–18] and Genetic Programming (GP).[19–22]

16.1.1. *Communications Data*

The first data set of interest in the present study is communications traffic — a total of 50,000 records — over a 2-year period. The data format is as shown in Table 16.2.

Data always exists for the first two columns in the record, but not necessarily with the third. Uni-directional data is the norm, although some bi-directional data is also contained within this data set. The anonymous data reflects communications activity over a single (tightly interconnected) network during the specified time period. Note that this does not necessarily correspond to people *per se*, but communications services. This network comprises a tightly-connected 'inner circle', with branches to a peripheral group, although some communications do emanate from outside this inner core. In other words, the data describes a social network of sorts.[23] Accordingly, in the next section we turn our attention to Social Network Analysis (SNA).

A second, more extensive (258,052 record), 2-year communications traffic data set was also made available during the course of this study (Table 16.3); it differs from the first data set in that the type of communication is also recorded.

16.2. Social Network Analysis

At its most fundamental level, Social Network Analysis (SNA) models 'entities' (such as people), and the relationships between them, as nodes and links

Table 16.2. Data Format #1.

ID	Date	Time & duration

Table 16.3. Data Format #2.

(sanitized) date & time	'from' service	'to' service

(respectively) on a graphical model.[24–26a] Such graphs can be likened to the entity-relationship (ER) diagrams commonly encountered in databases.[27–29]

While SNA emphasizes network structure, it also encompasses concepts such as 'between-ness', 'closeness', 'centrality' (placement of the most important/significant entities in the middle of graphs), 'clustering coefficient', 'cohesion', 'path length', 'radiality', 'reach', and the like.

Commonly encountered SNA software tools[30b] include `UCINet`,[c] `Pajek`,[31d] and the *network* software[e] developed for the `R` statistical analysis package.

SNA has been successfully applied to criminal investigations.[32,33]

16.3. Intelligent Reasoning Methods

Numerous approaches have been used for temporal data mining, including graph-based methods,[34,35] fractals,[36] Hidden Markov Models (HMMs),[37,38] Bayesian (Belief) networks,[37,39–41] transactional networks,[42] matrix decompositions,[43] and even Semantic Web-related techniques (e.g. RDF, RDFS and OWL).[44] In the present study we focus on the following techniques: (i) link mining, (ii) software agents, (iii) swarms, and (iv) Artificial Neural Networks.

16.3.1. *Link Mining*

Link (or relational) Mining is a specialized form of DM better suited to the mining of highly structured, heterogeneous data sets in which the objects (entities) are linked in some manner.[45] In graphical terms, objects (entities) correspond to nodes, whereas links (relationships) are represented by edges. 'Similarity' depends not only on comparing attributes but also on links (relationships) between objects (entities) — issues of familiarity and similarity are discussed further in Sects. 16.5.4 and 16.9, respectively. Link Mining has been driven by developments in fields as diverse as link analysis, graph mining, relational learning, inductive logic programming, but most especially hypertext and web mining. Link analysis is used in law enforcement, counter-intelligence, fraud detection and the like.[46] Fawcett and Provost investigated cellular (mobile) phone fraud.[47,48] It has the capability of *automatically* constructing link diagrams from large databases. A typical link analysis tool is `NetMap`.[49]

Link Mining has been applied to web page rankings produced by internet search engines, epidemiology (the spread of communicable diseases), and bibliography searching.

[a] http://www.orgnet.com/sna.html
[b] http://www.insna.org/INSNA/soft_inf.html
[c] http://www.analytictech.com/ucinet/ucinet.htm
[d] http://vlado.fmf.uni-lj.si/sub/networks/pajek/
[e] http://cran.r-project.org/src/contrtib/Descriptions/network.html

Typical Link Mining tasks include data classification (of entities over time), cluster analysis, identification of the number of links and their type, determination of link strengths, and the tracking of activities 'of interest'. However as Senator points out, no comprehensive framework yet exists for combining the link mining tasks commonly required for real-world applications; it remains an *ad hoc* activity.[50]

16.3.2. *Software Agents*

Software agents, as their name suggests, perform actions on behalf of their human 'owners'. They owe their origins to work in distributed AI during the 1980s.[51,52] Agents are autonomous, possess knowledge about their environment such that they are able to interact 'intelligently' with it, and are able to learn and adapt their behaviour over time.[5–7] Many different types of software agent have been developed over the years, a popular one being the so-called BDI (beliefs-desires-intentions) agent. A critical consideration with agents is the granting of permission to execute on remote hosts; oftentimes firewalls and other system security measures will prevent this by default, unless *explicit* permission is sought (and granted).

More specifically, software agents are characterized by the following:

(1) *persistence* — they run continuously and decide *for themselves* when they should perform various activities;
(2) *autonomy* — they are capable of task selection, prioritization, goal-directed behaviour, and decision making *without* human intervention;
(3) *social ability* — they are capable of *interacting* with other agents and/or humans in order to satisfy their design objectives;
(4) *reactivity* — they are able to *perceive* their environment, and *respond* in a timely fashion to changes that occur therein, in order to satisfy their design objectives.

Since the environment of interest — communications — is inherently complex and dynamic in nature, then agents are particularly well suited to the present study.

16.3.3. *Swarms*

Swarms were inspired initially by the flocking/herding/collective behaviour of social insects. The resulting 'collective intelligence' of the swarm (flock, herd) is able to achieve performance way beyond that of its constituent members, by virtue of the interactions between the (*un*-intelligent) individuals, for example through the laying down 'pheromone' trails.[53,54] Unlike evolutionary approaches,[12–14] it is the behaviour of *present*-generation members that determines swarm behaviour, not on the inherited characteristics ('genetic' information) passed on to successive population generations. Nevertheless, the two approaches are inherently iterative, data-driven and bottom-up in nature. Several (open source) packages are in widespread

(respectively) on a graphical model.[24–26a] Such graphs can be likened to the entity-relationship (ER) diagrams commonly encountered in databases.[27–29]

While SNA emphasizes network structure, it also encompasses concepts such as 'between-ness', 'closeness', 'centrality' (placement of the most important/significant entities in the middle of graphs), 'clustering coefficient', 'cohesion', 'path length', 'radiality', 'reach', and the like.

Commonly encountered SNA software tools[30b] include `UCINet`,[c] `Pajek`,[31d] and the *network* software[e] developed for the `R` statistical analysis package.

SNA has been successfully applied to criminal investigations.[32,33]

16.3. Intelligent Reasoning Methods

Numerous approaches have been used for temporal data mining, including graph-based methods,[34,35] fractals,[36] Hidden Markov Models (HMMs),[37,38] Bayesian (Belief) networks,[37,39–41] transactional networks,[42] matrix decompositions,[43] and even Semantic Web-related techniques (e.g. RDF, RDFS and OWL).[44] In the present study we focus on the following techniques: (i) link mining, (ii) software agents, (iii) swarms, and (iv) Artificial Neural Networks.

16.3.1. *Link Mining*

Link (or relational) Mining is a specialized form of DM better suited to the mining of highly structured, heterogeneous data sets in which the objects (entities) are linked in some manner.[45] In graphical terms, objects (entities) correspond to nodes, whereas links (relationships) are represented by edges. 'Similarity' depends not only on comparing attributes but also on links (relationships) between objects (entities) — issues of familiarity and similarity are discussed further in Sects. 16.5.4 and 16.9, respectively. Link Mining has been driven by developments in fields as diverse as link analysis, graph mining, relational learning, inductive logic programming, but most especially hypertext and web mining. Link analysis is used in law enforcement, counter-intelligence, fraud detection and the like.[46] Fawcett and Provost investigated cellular (mobile) phone fraud.[47,48] It has the capability of *automatically* constructing link diagrams from large databases. A typical link analysis tool is `NetMap`.[49]

Link Mining has been applied to web page rankings produced by internet search engines, epidemiology (the spread of communicable diseases), and bibliography searching.

[a] http://www.orgnet.com/sna.html
[b] http://www.insna.org/INSNA/soft_inf.html
[c] http://www.analytictech.com/ucinet/ucinet.htm
[d] http://vlado.fmf.uni-lj.si/sub/networks/pajek/
[e] http://cran.r-project.org/src/contrtib/Descriptions/network.html

Typical Link Mining tasks include data classification (of entities over time), cluster analysis, identification of the number of links and their type, determination of link strengths, and the tracking of activities 'of interest'. However as Senator points out, no comprehensive framework yet exists for combining the link mining tasks commonly required for real-world applications; it remains an *ad hoc* activity.[50]

16.3.2. *Software Agents*

Software agents, as their name suggests, perform actions on behalf of their human 'owners'. They owe their origins to work in distributed AI during the 1980s.[51,52] Agents are autonomous, possess knowledge about their environment such that they are able to interact 'intelligently' with it, and are able to learn and adapt their behaviour over time.[5–7] Many different types of software agent have been developed over the years, a popular one being the so-called BDI (beliefs-desires-intentions) agent. A critical consideration with agents is the granting of permission to execute on remote hosts; oftentimes firewalls and other system security measures will prevent this by default, unless *explicit* permission is sought (and granted).

More specifically, software agents are characterized by the following:

(1) *persistence* — they run continuously and decide *for themselves* when they should perform various activities;
(2) *autonomy* — they are capable of task selection, prioritization, goal-directed behaviour, and decision making *without* human intervention;
(3) *social ability* — they are capable of *interacting* with other agents and/or humans in order to satisfy their design objectives;
(4) *reactivity* — they are able to *perceive* their environment, and *respond* in a timely fashion to changes that occur therein, in order to satisfy their design objectives.

Since the environment of interest — communications — is inherently complex and dynamic in nature, then agents are particularly well suited to the present study.

16.3.3. *Swarms*

Swarms were inspired initially by the flocking/herding/collective behaviour of social insects. The resulting 'collective intelligence' of the swarm (flock, herd) is able to achieve performance way beyond that of its constituent members, by virtue of the interactions between the (*un*-intelligent) individuals, for example through the laying down 'pheromone' trails.[53,54] Unlike evolutionary approaches,[12–14] it is the behaviour of *present*-generation members that determines swarm behaviour, not on the inherited characteristics ('genetic' information) passed on to successive population generations. Nevertheless, the two approaches are inherently iterative, data-driven and bottom-up in nature. Several (open source) packages are in widespread

use, one of the more popular being the (open source, java-based) Recursive Porous Agent Simulation Toolkit — `RePast`.[f]

Of particular interest to the present project is the fact that swarms have been widely applied in agent-based simulations (ABSs) in the past,[55,56] and hold potential for tracking both individual and group life cycles. In other words, they are essentially dynamic, and hence well suited to the modelling/display of how social networks change over time (in contrast, say, to the *static* network displays produced by `JUNG`); this capability is explored further in Sect. 16.7.2.

16.3.4. *Artificial Neural Networks*

Like swarms, Artificial Neural Networks (ANNs) take their inspiration from Nature. They are simplified models of biological neural networks ('brains'), comprising 'neurons' (nodes) along with interconnecting weights, which are trained using representative input-output exemplars.[57–59] Rather than being a model-driven, algorithmic approach, they constitute a bottom-up, data-driven one.[3] As with Data Mining (Sect. 16.1), pre-processing is critical for good performance — even more so, since ANNs typically require *many* training iterations to arrive at an acceptable solution (corresponding to a global minimum in the energy/solution landscape). The input patterns are associated with their corresponding output pairs within the network weights (but in a holistic, rather than in a one-to-one sense). Despite their long training times, ANNs are able to respond to new input data (patterns) immediately, indeed they are well known to excel at pattern recognition, and this is the reason for their utilization in the *Knowledge Discovery* Module of our MAS-based System model.

16.4. Multi-Agent System (MAS) Network Model

Figure 16.1 shows our overall system block diagram. Although we chose a Multi-Agent framework, our design is not limited to MASs — as indicated in the *Knowledge Discovery* Module — rather a range of techniques can be accommodated under this umbrella, including (but not restricted to):

- Link mining,
- Correlation matrix,
- Markov Chains (for prediction purposes),
- Swarms,
- Artificial Neural Networks (as outlined in Sect. 16.3.4),
- Evolutionary Algorithms,
- Machine Learning techniques.

[f] http://repast.sourceforge.net/

Typical Link Mining tasks include data classification (of entities over time), cluster analysis, identification of the number of links and their type, determination of link strengths, and the tracking of activities 'of interest'. However as Senator points out, no comprehensive framework yet exists for combining the link mining tasks commonly required for real-world applications; it remains an *ad hoc* activity.[50]

16.3.2. *Software Agents*

Software agents, as their name suggests, perform actions on behalf of their human 'owners'. They owe their origins to work in distributed AI during the 1980s.[51,52] Agents are autonomous, possess knowledge about their environment such that they are able to interact 'intelligently' with it, and are able to learn and adapt their behaviour over time.[5–7] Many different types of software agent have been developed over the years, a popular one being the so-called BDI (beliefs-desires-intentions) agent. A critical consideration with agents is the granting of permission to execute on remote hosts; oftentimes firewalls and other system security measures will prevent this by default, unless *explicit* permission is sought (and granted).

More specifically, software agents are characterized by the following:

(1) *persistence* — they run continuously and decide *for themselves* when they should perform various activities;
(2) *autonomy* — they are capable of task selection, prioritization, goal-directed behaviour, and decision making *without* human intervention;
(3) *social ability* — they are capable of *interacting* with other agents and/or humans in order to satisfy their design objectives;
(4) *reactivity* — they are able to *perceive* their environment, and *respond* in a timely fashion to changes that occur therein, in order to satisfy their design objectives.

Since the environment of interest — communications — is inherently complex and dynamic in nature, then agents are particularly well suited to the present study.

16.3.3. *Swarms*

Swarms were inspired initially by the flocking/herding/collective behaviour of social insects. The resulting 'collective intelligence' of the swarm (flock, herd) is able to achieve performance way beyond that of its constituent members, by virtue of the interactions between the (*un*-intelligent) individuals, for example through the laying down 'pheromone' trails.[53,54] Unlike evolutionary approaches,[12–14] it is the behaviour of *present*-generation members that determines swarm behaviour, not on the inherited characteristics ('genetic' information) passed on to successive population generations. Nevertheless, the two approaches are inherently iterative, data-driven and bottom-up in nature. Several (open source) packages are in widespread

use, one of the more popular being the (open source, java-based) Recursive Porous Agent Simulation Toolkit — `RePast`.[f]

Of particular interest to the present project is the fact that swarms have been widely applied in agent-based simulations (ABSs) in the past,[55,56] and hold potential for tracking both individual and group life cycles. In other words, they are essentially dynamic, and hence well suited to the modelling/display of how social networks change over time (in contrast, say, to the *static* network displays produced by `JUNG`); this capability is explored further in Sect. 16.7.2.

16.3.4. *Artificial Neural Networks*

Like swarms, Artificial Neural Networks (ANNs) take their inspiration from Nature. They are simplified models of biological neural networks ('brains'), comprising 'neurons' (nodes) along with interconnecting weights, which are trained using representative input-output exemplars.[57–59] Rather than being a model-driven, algorithmic approach, they constitute a bottom-up, data-driven one.[3] As with Data Mining (Sect. 16.1), pre-processing is critical for good performance — even more so, since ANNs typically require *many* training iterations to arrive at an acceptable solution (corresponding to a global minimum in the energy/solution landscape). The input patterns are associated with their corresponding output pairs within the network weights (but in a holistic, rather than in a one-to-one sense). Despite their long training times, ANNs are able to respond to new input data (patterns) immediately, indeed they are well known to excel at pattern recognition, and this is the reason for their utilization in the *Knowledge Discovery* Module of our MAS-based System model.

16.4. Multi-Agent System (MAS) Network Model

Figure 16.1 shows our overall system block diagram. Although we chose a Multi-Agent framework, our design is not limited to MASs — as indicated in the *Knowledge Discovery* Module — rather a range of techniques can be accommodated under this umbrella, including (but not restricted to):

- Link mining,
- Correlation matrix,
- Markov Chains (for prediction purposes),
- Swarms,
- Artificial Neural Networks (as outlined in Sect. 16.3.4),
- Evolutionary Algorithms,
- Machine Learning techniques.

[f] http://repast.sourceforge.net/

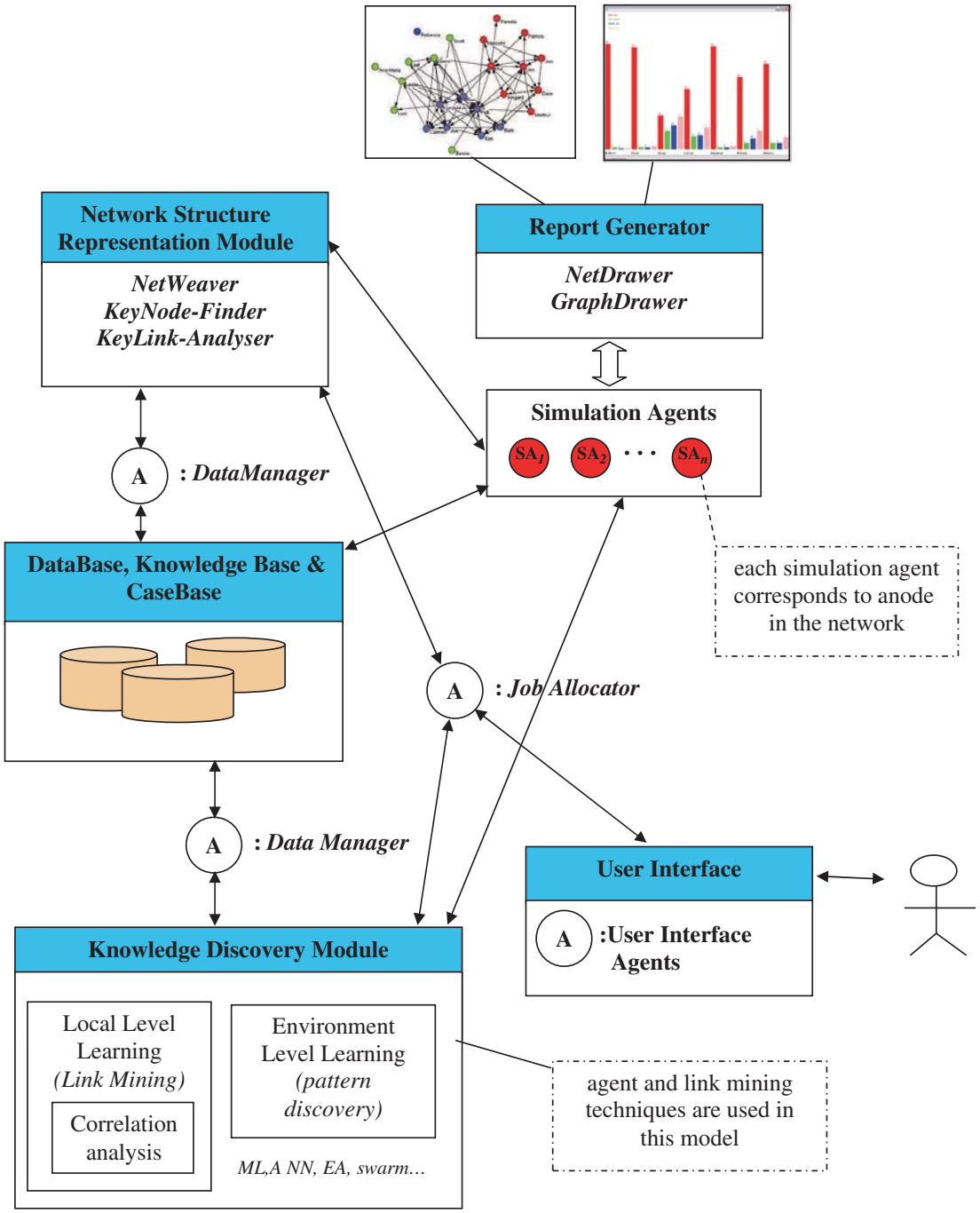

Fig. 16.1. System Block Diagram.

For instance, there may well be benefit in combining agent-based learning (a local technique) with Data Mining (a global technique) — more specifically, in order to discover more about relationships between nodes (accounts, people and phone owners), changing trends, and other previously undiscovered information/knowledge.

Agent learning could proceed on two levels:

(1) system (*based on time*), and
(2) local (*based on traffic*).

use, one of the more popular being the (open source, java-based) Recursive Porous Agent Simulation Toolkit — `RePast`.[f]

Of particular interest to the present project is the fact that swarms have been widely applied in agent-based simulations (ABSs) in the past,[55,56] and hold potential for tracking both individual and group life cycles. In other words, they are essentially dynamic, and hence well suited to the modelling/display of how social networks change over time (in contrast, say, to the *static* network displays produced by `JUNG`); this capability is explored further in Sect. 16.7.2.

16.3.4. *Artificial Neural Networks*

Like swarms, Artificial Neural Networks (ANNs) take their inspiration from Nature. They are simplified models of biological neural networks ('brains'), comprising 'neurons' (nodes) along with interconnecting weights, which are trained using representative input-output exemplars.[57–59] Rather than being a model-driven, algorithmic approach, they constitute a bottom-up, data-driven one.[3] As with Data Mining (Sect. 16.1), pre-processing is critical for good performance — even more so, since ANNs typically require *many* training iterations to arrive at an acceptable solution (corresponding to a global minimum in the energy/solution landscape). The input patterns are associated with their corresponding output pairs within the network weights (but in a holistic, rather than in a one-to-one sense). Despite their long training times, ANNs are able to respond to new input data (patterns) immediately, indeed they are well known to excel at pattern recognition, and this is the reason for their utilization in the *Knowledge Discovery* Module of our MAS-based System model.

16.4. Multi-Agent System (MAS) Network Model

Figure 16.1 shows our overall system block diagram. Although we chose a Multi-Agent framework, our design is not limited to MASs — as indicated in the *Knowledge Discovery* Module — rather a range of techniques can be accommodated under this umbrella, including (but not restricted to):

- Link mining,
- Correlation matrix,
- Markov Chains (for prediction purposes),
- Swarms,
- Artificial Neural Networks (as outlined in Sect. 16.3.4),
- Evolutionary Algorithms,
- Machine Learning techniques.

[f] http://repast.sourceforge.net/

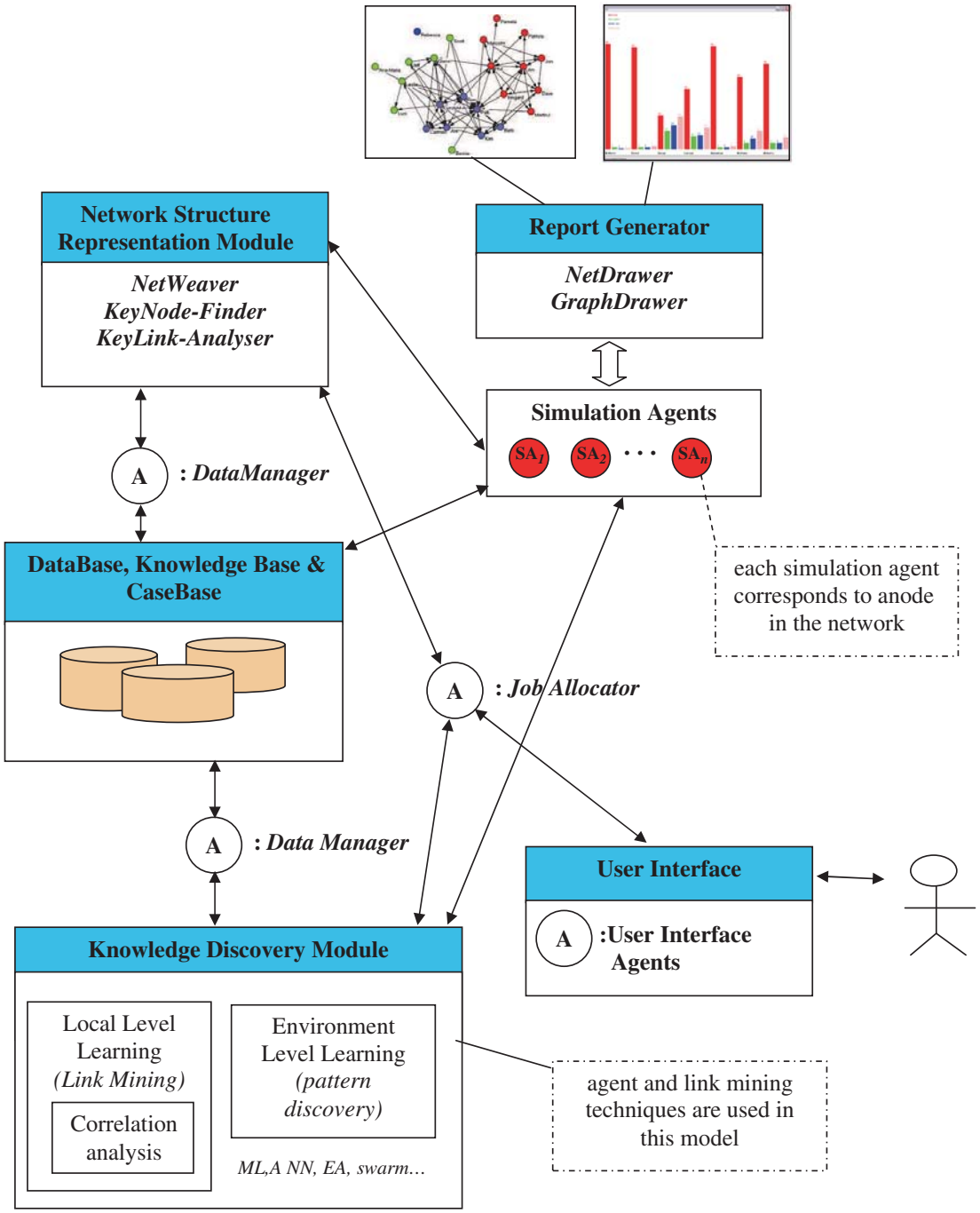

Fig. 16.1. System Block Diagram.

For instance, there may well be benefit in combining agent-based learning (a local technique) with Data Mining (a global technique) — more specifically, in order to discover more about relationships between nodes (accounts, people and phone owners), changing trends, and other previously undiscovered information/knowledge.

Agent learning could proceed on two levels:

(1) system (*based on time*), and
(2) local (*based on traffic*).

The first would focus on how both traffic and links between nodes change over time, while the second would concentrate on correlations between nodes. A combination of agents and link mining (or other data mining method) could prove useful with the former, while a combination of agents and pattern matching (or other Machine Learning method) may prove useful with the latter.

We invoked three different types of agent facilitators in order to realize our social network MAS model, these being (i) UserInterface, (ii) JobAllocator, and (iii) DataManager. The *UserInterface* agent receives and pre-processes user requests, prior to forwarding these on to the *JobAllocator*. *JobAlloactor* is a middle-level agent that allocates user requests to the most appropriate model. The *DataManager* agent manages the data contained within the data/knowledge/case bases (*Network Structure Representation, Correlation Analysis* and *Report Generator* modules), and supplies data to the upper-level module(s) appropriate for specific user requests.

The Network Structure Representation module incorporates *NetWeaver, KeyNodeFinder* and *KeyLinkAnalyser* agents:

- *NetWeaver* reads raw data (.txt files) and performs any necessary pre-processing. More specifically, it converts data files into appropriate SNA format (e.g. .net(work) files in Pajek format, as used as inputs in `Jung`). *NetWeaver* is able to receive user requests from the *JobAllocator* and to provide the required .net files to *NetDrawer* (in the *Report Generator* module). *NetWeaver* also provides data to the *DataBase* module for historical analysis.
- *KeyNodeFinder* identifies 'important' network nodes and sends the KeyNode list to the *DataBase* and *Report Generation* modules; currently 'important' is characterized by (i) heavy call traffic, (ii) the node in question being a 'key' node ('hub'), (iii) user-selected, or (iv) the largest number of links.
- *KeyLinkAnalyser* identifies and analyzes 'important' links between nodes, and provides results to the *DataBase* and *Report Generation* modules.

The *Correlation Analysis* module comprises both *HighCorrelationFinder* and *PairCorrelationAnalyzer* agents. The former identifies pairs with high correlation coefficients and reports results to the *DataBase* and *Report Generator* modules, whereas the latter identifies correlation coefficients between user-specified node pairs.

In the *Report Generator* module, *NetDrawer* receives data from *NetWeaver* and displays the corresponding network graph; *GraphWeaver* produces correlation graphs based on data received from the *Correlation Analysis* module(s).

Initially, `AgentBuilder`[g] was used in the development of the above modules (and agents), but subsequently Java proved less restrictive. A general-purpose agent was used at first, but later specialized agents were employed for each respective module. Initially we created 'agent' classes as 'beliefs' within `AgentBuilder`, in which

[g]http://www.agentbuilder.com/

the user selects 'ontology' — i.e. via `NetShow` — which implies classes/methods (in other words, instances of objects); subsequently we utilized Java classes.

16.5. `NetShow` Software

During the course of this study we developed software for the automatic extraction of network descriptors, followed by display of the social network under study, using the system model of Figure 16.1. If we are able to model the communications network with sufficient accuracy, then this should facilitate the 'reverse-engineering' of data leading up to some external event, in order to predict 'similar' occurrences in the future, but note that this pre-supposes the ready availability of representative case studies (we will return to consider this issue in Sec. 16.8). Our longer-term goal is therefore to develop an agent-based framework for first modeling then ultimately predicting the behaviour of the social network of interest. To this end, call frequency is deemed to be more significant than call duration. Accordingly, we display call frequencies (for caller-callee pairs) as the interconnection/link strengths (magnitudes) between nodes in the SNA graph.

`NetShow` automatically extracts the following dynamic descriptors: network architecture, links, traffic, key node analysis and correlation coefficients. Figure 16.2 shows a typical display produced by `NetShow`.

It becomes immediately apparent that there are two 'central' nodes in this particular network, these being ON19074N6A and OA2N05N855; likewise the highest call traffic (link strengths) are 76.0 (0746A287N7 → 0746A9619N), 72.0 (0A2557458N → 0A2N796N19), 56.0 (0A2N197052 → 0N19074N6A), and so on. Network pruning (say by removing all links less than 5.0 in strength) renders this even more apparent.

Moreover, since `NetShow` was developed using `UCINet`, it is capable of performing classical 'cut point' SNA. For example, the following 12 'k-cores'[h] were automatically extracted from Figure 16.3, where the link strengths have been removed for improved readability (but are still available within the program): 0A2N05N855, 0746505N78, 07465N5N19, 0A2N8NN702, 0N19074587, 0A2N877N7N, 0A2N858-N6A, 0A2N197052, 0A2N19A052, 0A2N5357N7, 0N19074N6A, 0A2N796N19, 0A2N750687, and 07465011A6.

The Java Universal Network Graphical package (`JUNG`)[i] was also used to facilitate the development of `NetShow`, however the latter is capable of displaying not only static 'snapshots' of the network, but also dynamic displays of network architecture and link strengths, with screen updates taking place every 30–60 seconds.

[h]Given a graph $G = \{V, E\}$ with vertices set V and edges set E, the k-core is computed by pruning all vertices (along with their respective edges) with degree (that is, number of connections to other nodes) less than k.

[i]http://jung.sourceforge.net/

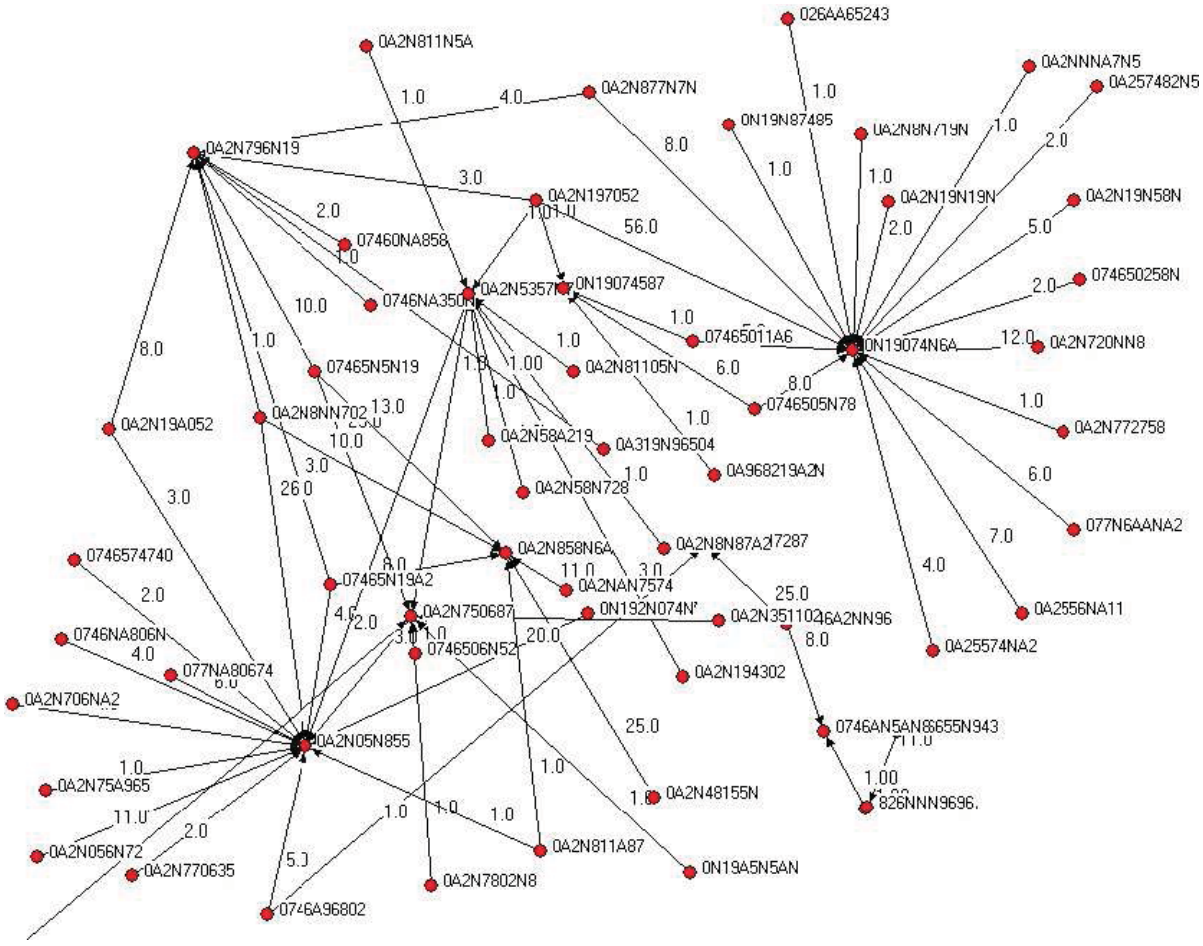

Fig. 16.2. Network Node Diagram (over a 7-month period).

16.5.1. NetShow 2.0 *Software*

The original NetShow program was expanded and refined, with NetShow 2.0 being completely re-written using more recent releases of both Java (V1.5) and JUNG (V2.0), leading to the following improvements:

Figure 16.4 shows how users are able to create images from the current graph within the NetShow 2.0 GUI, while Figure 16.5 shows the automatic report generator in action (i) through (iv).

(i) it is now linked with a *bona fide* database (MySQL) rather than reading data records from a .txt file;
(ii) it can now automatically generate report files, which can include network structure graphs, related network analysis results, as well as a log of all operations performed by the user (refer to the leftmost panel in the ensuing screen dumps);
(iii) it allows users to select nodes and include related information about them in reports;
(iv) visualization functions have been greatly improved, courtesy of new functions incorporated into JUNG V2.0.

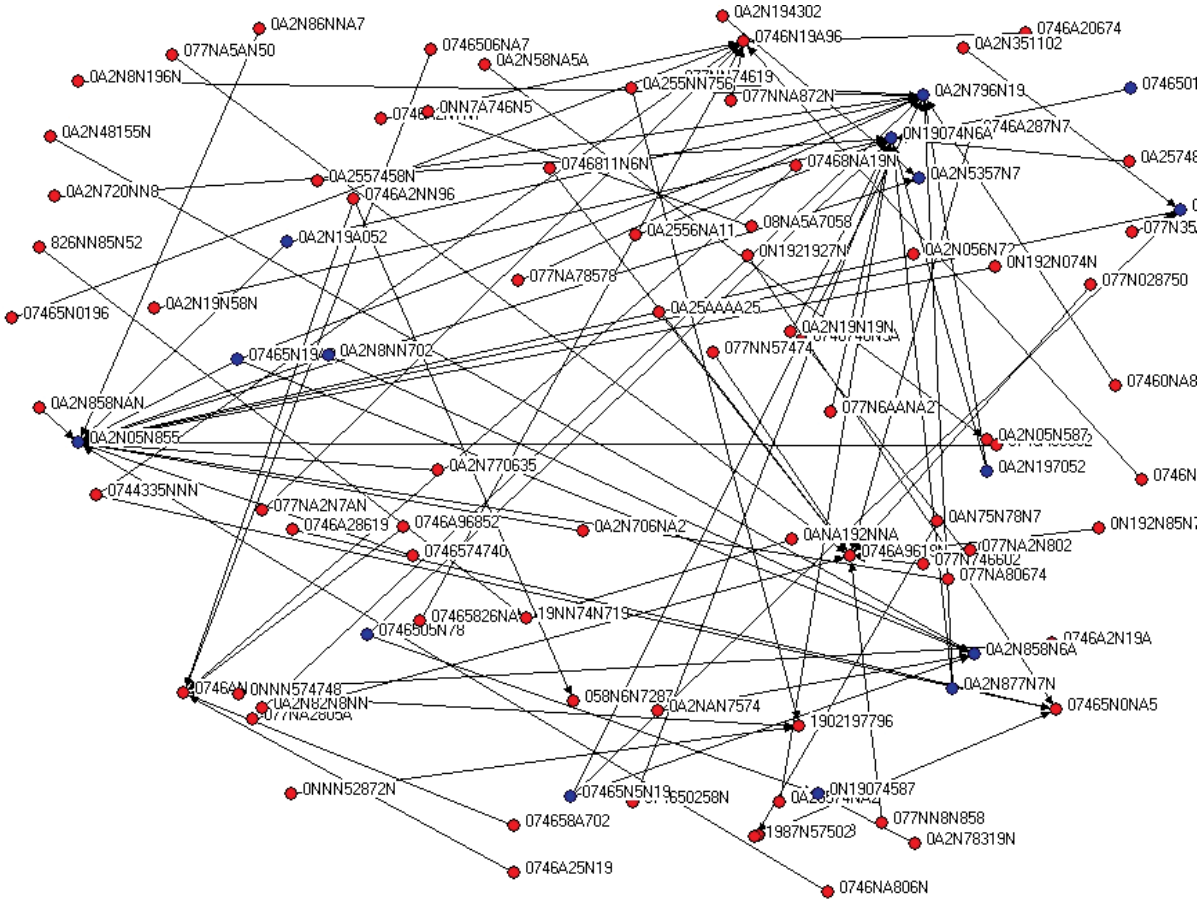

Fig. 16.3. Network Node Diagram (2002; link weights >2.0).

Since the database contains two different types of data (telecommunications traffic), we need to utilize two different kinds of agents. However in order to cater for future revision and/or expansion, both need to be based on an abstract class in which all common attributes and functions are defined. Once generated, an agent needs to possess the following capability:

(1) read data from the database and fill the agent's attributes accordingly,
(2) analyze the data, summarize the current situation and predict possible future scenarios,
(3) communicate with other agents, sharing data and analyses where appropriate (and if permitted), and
(4) visualize and represent analysis results (in various formats/styles, as appropriate).

16.5.2. *Network Visualization*

`JUNG V2.0` supports different presentation/display options, including tree, radial, balloon, label, minimum spanning, FR2 and (preferred) isometric (Figure 16.6).

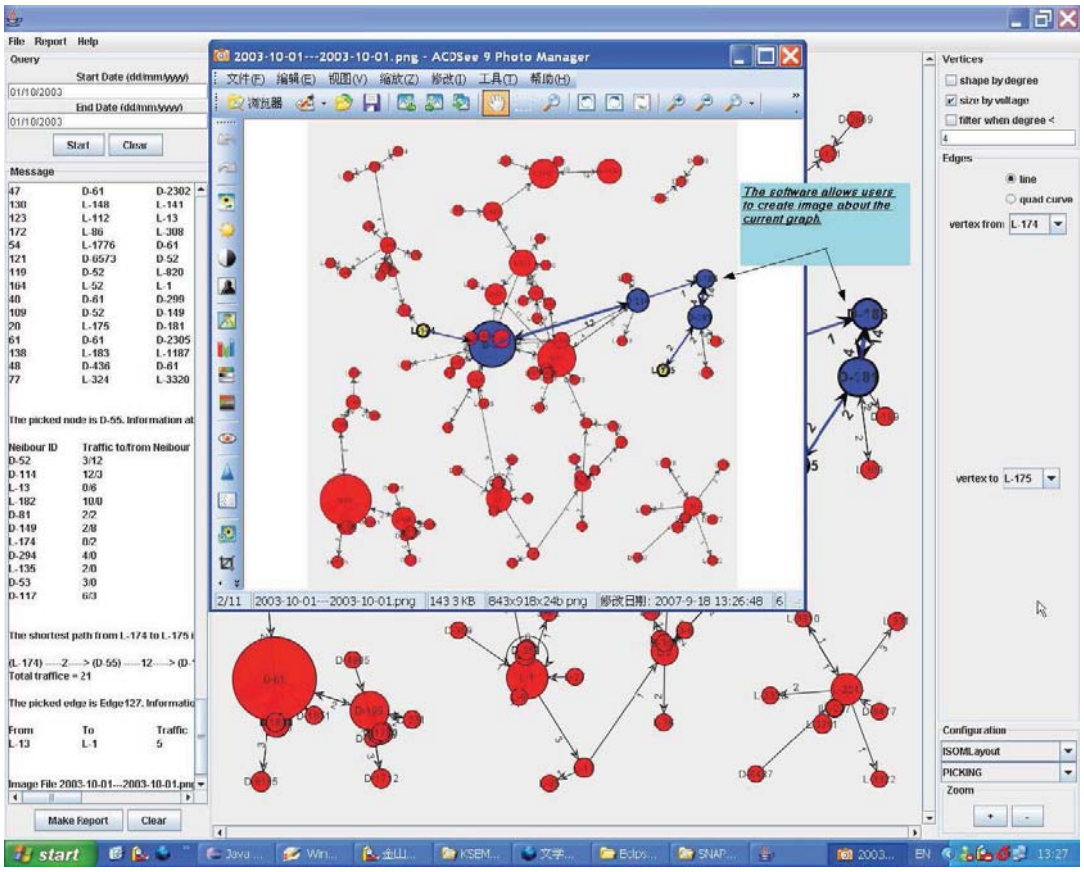

Fig. 16.4. Image Creation.

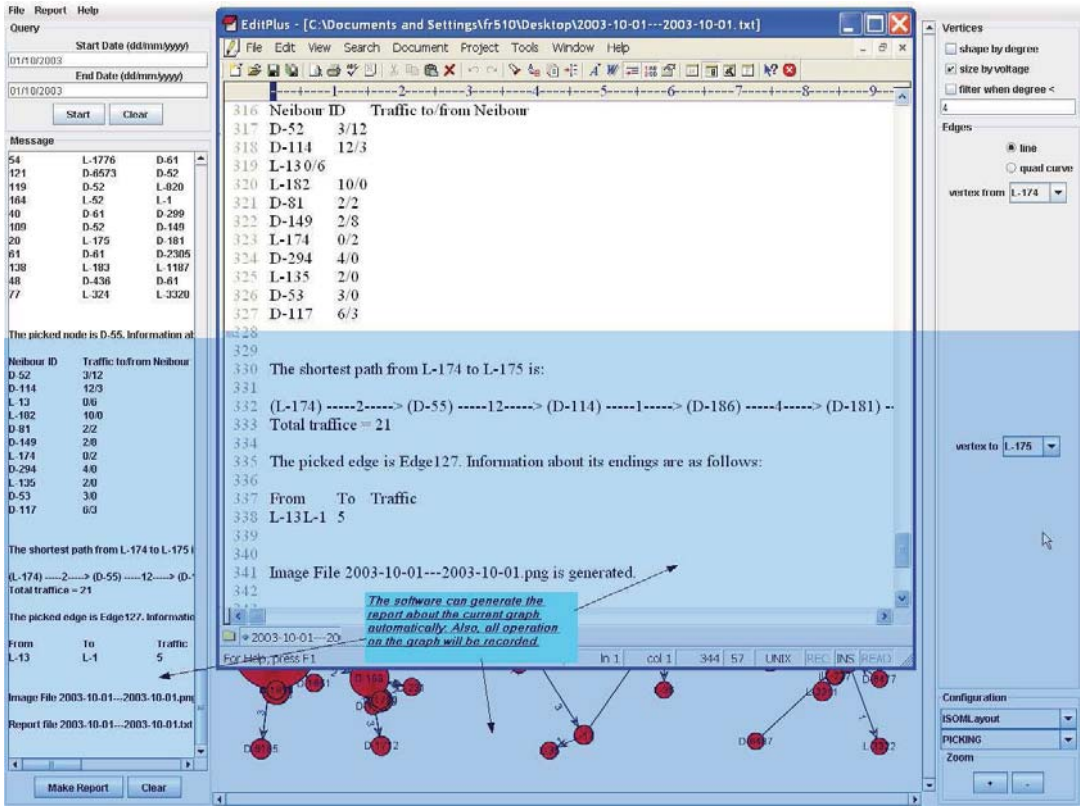

Fig. 16.5. Automatic Report Generation.

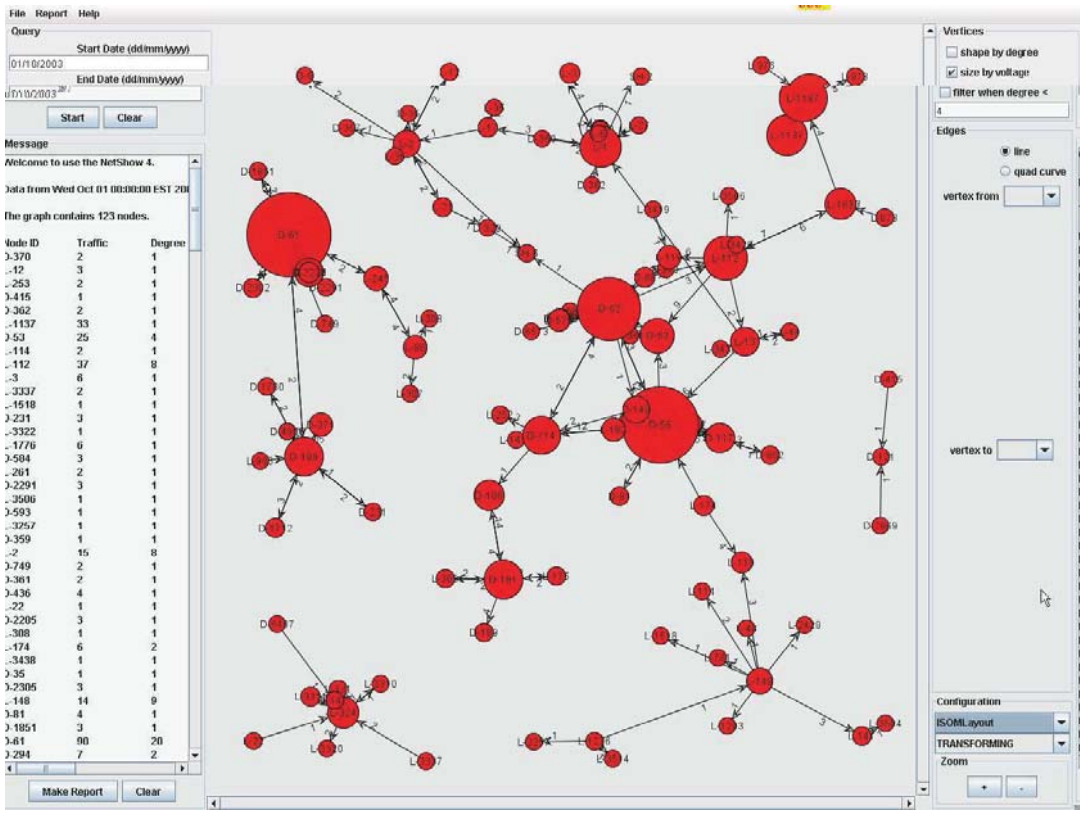

Fig. 16.6. ISOMLayout.

Visualization functions within `NetShow 2.0` have been improved as follows:

- it uses node size and shape to demonstrate both the degree of the node and the traffic passing through it (Figure 16.7):
  (a) nodes with larger node sizes have higher traffic;
  (b) nodes with more shape edges have higher degrees.
- it uses line thickness to represent link weights (Figure 16.8);
- it allows users to select a node, and to highlight that particular node's neighbours (Figure 16.9);
- it can show the 'shortest path' between two user-selected nodes. Furthermore, the shortest path will be highlighted and related path information included in the report (Figure 16.10).

16.5.3. *Pattern Discovery from Contact Lists*

We now present an example of how `NetShow 2.0` can be used for pattern discovery. Contact Lists for individual network nodes can be generated, then pattern analysis performed. It is also possible to use Association Rules (ARs) to analyze contact associations of individual nodes, as follows:

(1) first convert the daily contacts of the node in question into a transaction,
(2) then discover ARs from the node's transactions.

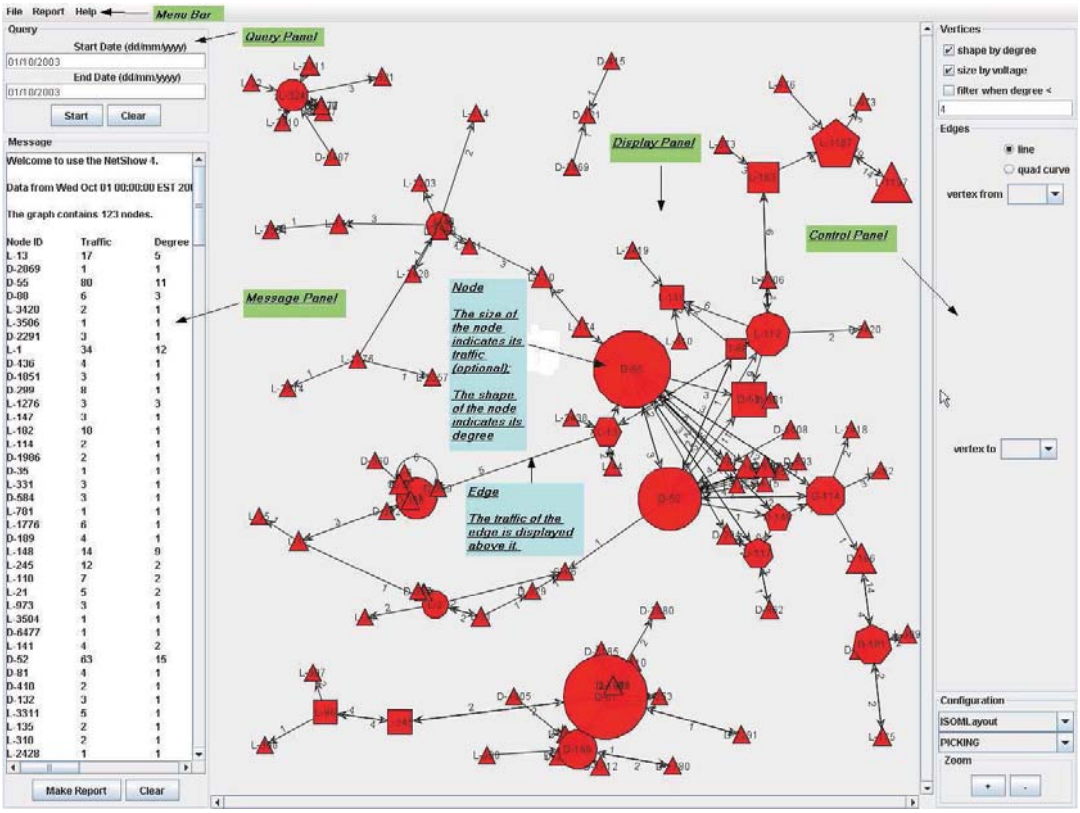

Fig. 16.7. Node Size and Shape.

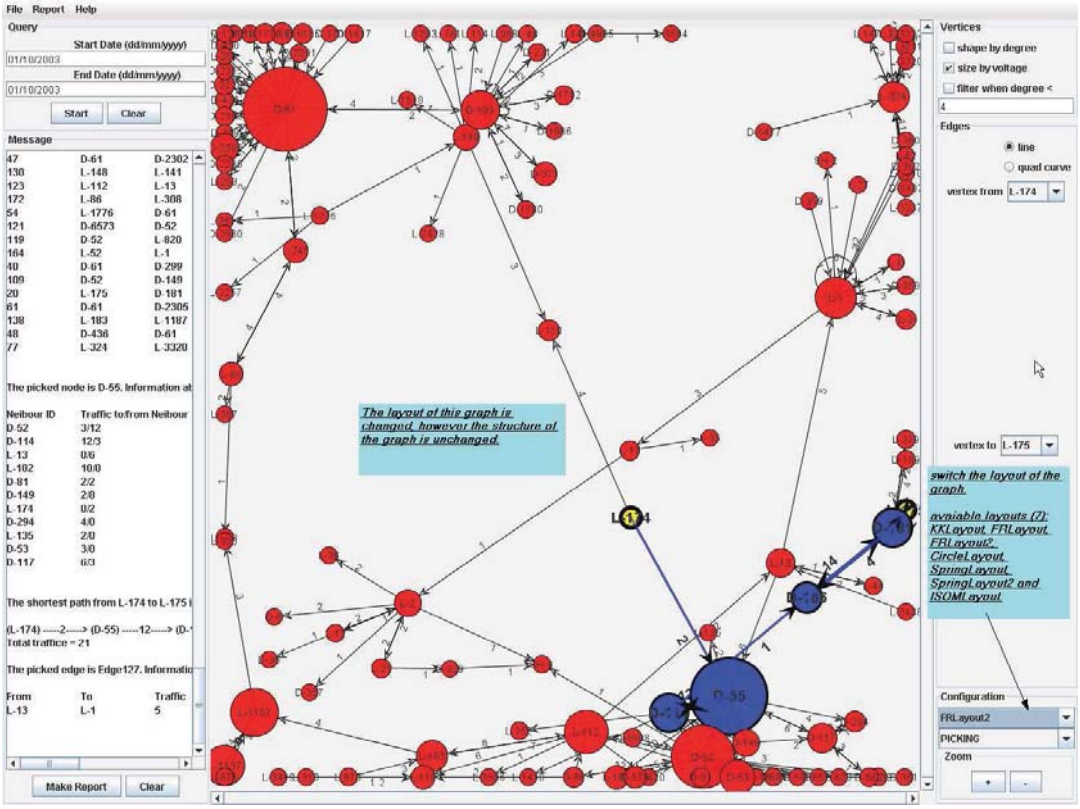

Fig. 16.8. Link Weights.

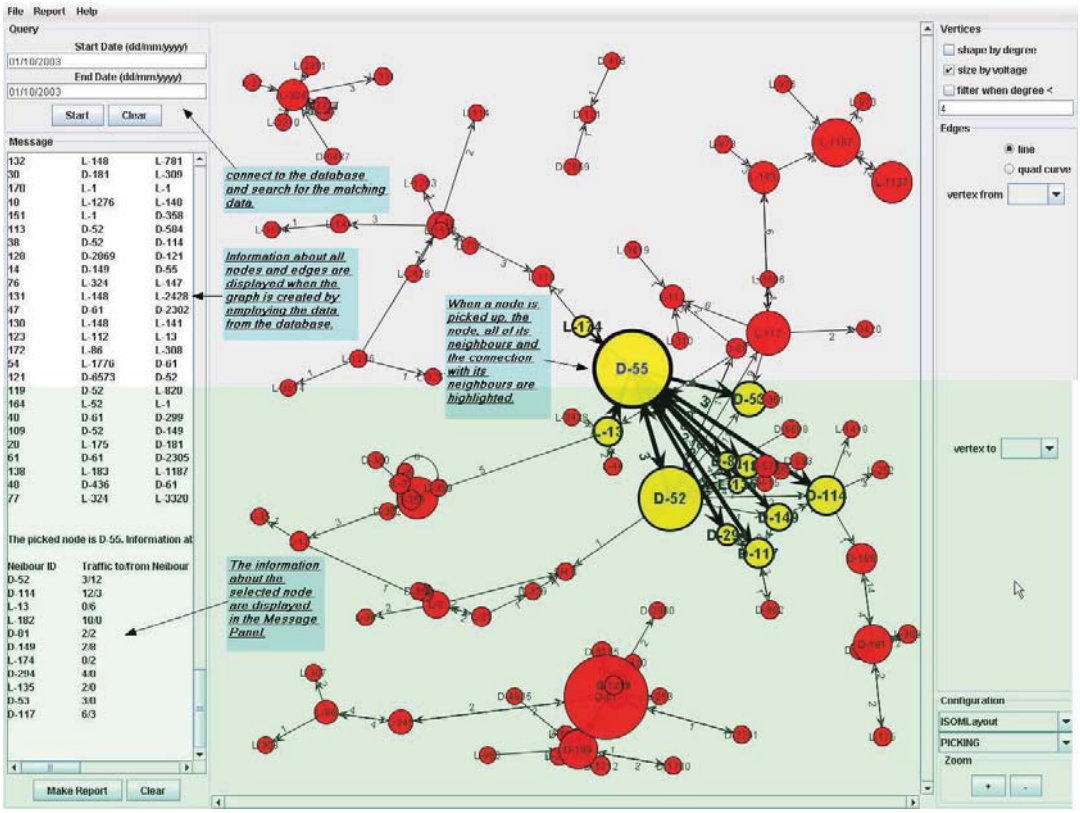

Fig. 16.9. Neighbours.

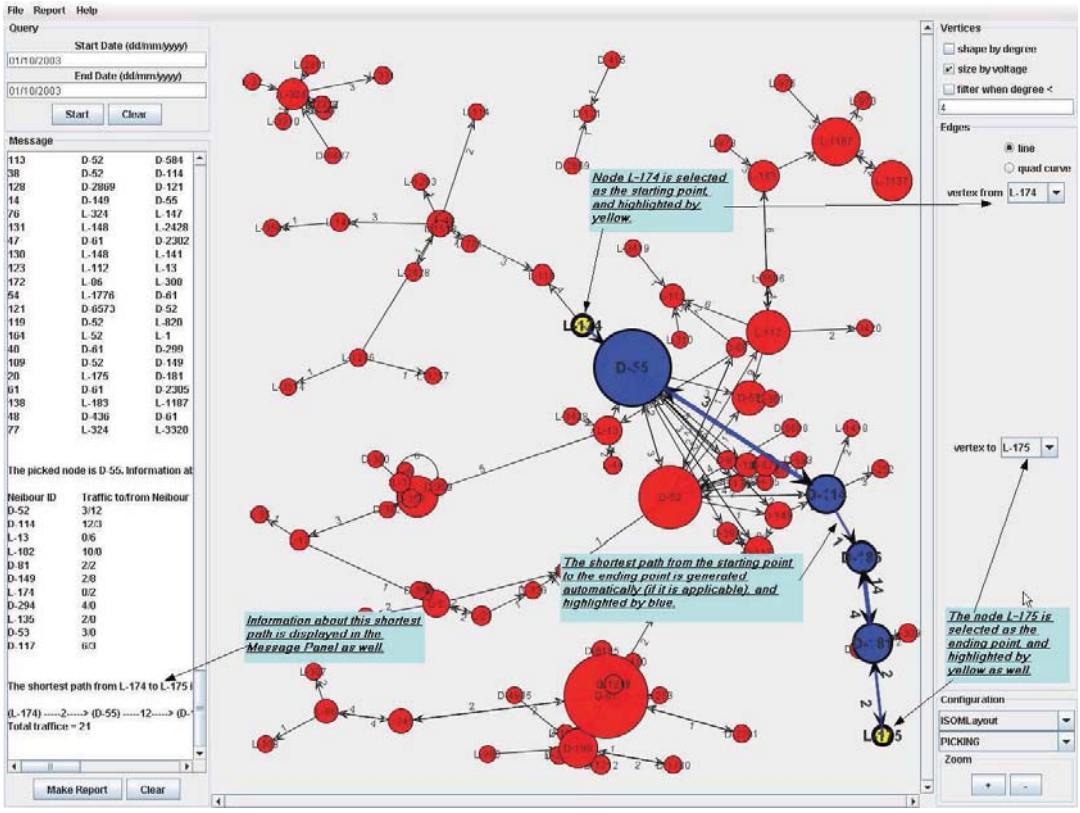

Fig. 16.10. Shortest Path.

Table 16.4. Tracking Familiarity Over Time.

January 2000		February 2000	
Contact	Familiarity	Contact	Familiarity
A	0.57	A	0.64
B	0.23	B	0.20
C	0.15	C	0.16
D	0.05	D	0.10

Using this method, we can expect to discover associations between the contacts of a particular node. In addition, we can incorporate some period selection processes before conducting the aforementioned AR analysis. In this manner, we are able to determine contact associations during a nominated time interval.

16.5.4. *Familiarity Analysis*

As a second illustration of the usefulness of `NetShow 2.0`, let us consider the issue of 'familiarity'. As already observed, each network node has an associated *Contact List*. The frequency and traffic between two nodes can be used to first define the degree-of-familiarity between two nodes, then this information added to the node's *Contact List*. The *Contact List* can then be sorted according to familiarity degree. In this manner, we can track changes in the relationships between nodes and their contacts. Table 16.4 shows an example of tracking familiarity over time.

16.6. Communications Analysis

16.6.1. *First Communications Data Set*

In order to characterize the social network of interest, ideally we would like data to be made available for a reference population, in order to compare and contrast the behaviour(s) of our group with 'standard' communications patterns; to couch this in immunity-based computing (IBC) terminology: *self* versus *non-self*.[60] For instance, a typical person will tend to make regular calls to friends, family, and businesses (and most likely with this order of frequency). They will also typically opt for economical telephone services (be that via a plan or using a pre-paid service). Lastly, they will both expect and deliver quick responses (witness the rapid turnaround times with which teenagers text each other).

Given the above, any investigation of the data will necessarily be looking for anomalous patterns over different time frames — in other words, daily/weekly/seasonal (and similar) variations.

A preliminary static analysis of this first telephone call traffic data set revealed a high proportion of short-duration (<30 second) calls. Next, the ten most frequent callers during a 6-month period were identified. The top four of these are shown in Figure 16.11, with traffic split into bi-directional links where known.

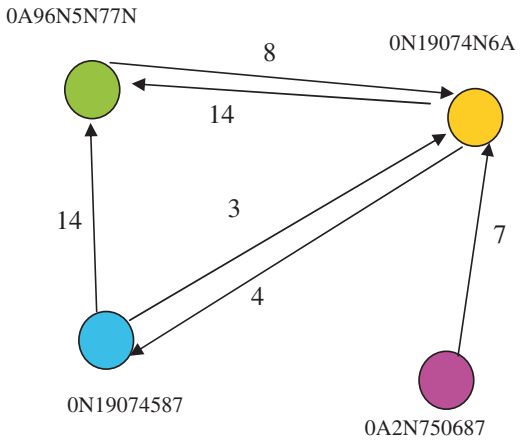

Fig. 16.11. Link Diagram (January–September 2004).

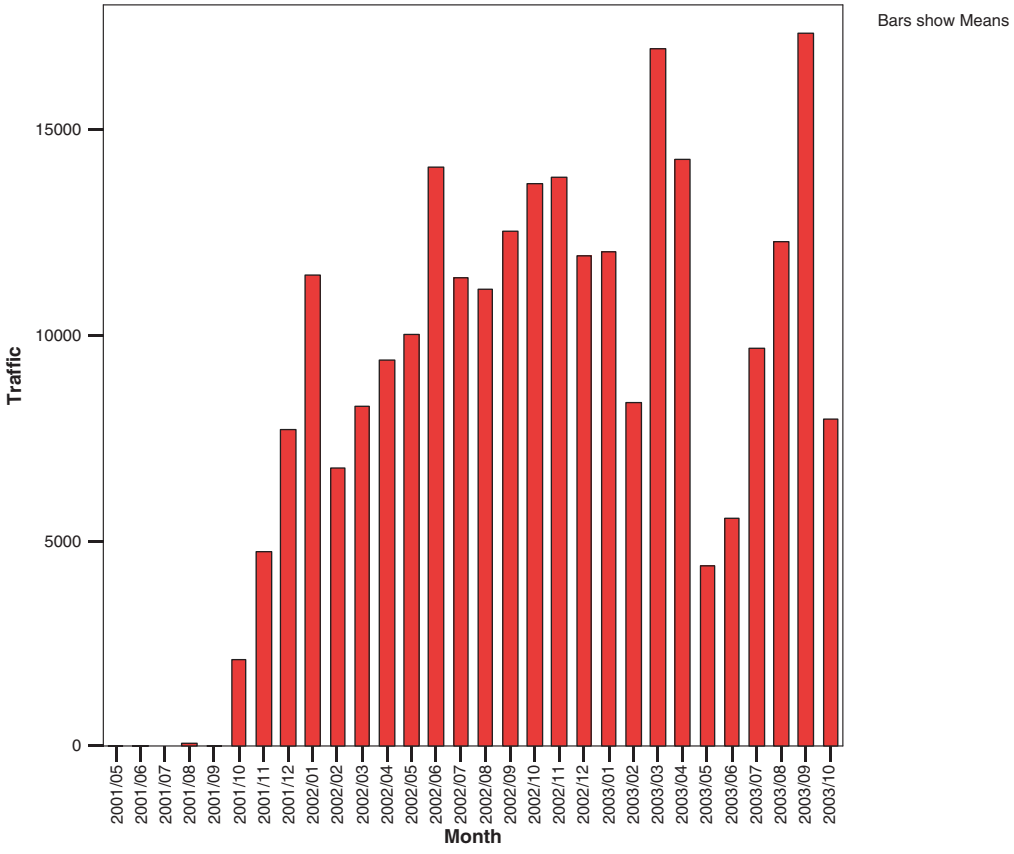

Fig. 16.12. Communications Traffic (24-month period).

Dynamic data analysis is of more interest than static however. In keeping with our own admonition *viz-a-viz* to visualize the data prior to invoking DM techniques in earnest, we simply plot the total traffic over the 2-year period, looking for general patterns of activity/inactivity. Figure 16.12 shows a gradual/steady increase during the first 12–14 months, with peaks at January and June of the first year, as well as March and September of the second year, together with a trough at May of

Table 16.4. Tracking Familiarity Over Time.

January 2000		February 2000	
Contact	Familiarity	Contact	Familiarity
A	0.57	A	0.64
B	0.23	B	0.20
C	0.15	C	0.16
D	0.05	D	0.10

Using this method, we can expect to discover associations between the contacts of a particular node. In addition, we can incorporate some period selection processes before conducting the aforementioned AR analysis. In this manner, we are able to determine contact associations during a nominated time interval.

16.5.4. *Familiarity Analysis*

As a second illustration of the usefulness of `NetShow 2.0`, let us consider the issue of 'familiarity'. As already observed, each network node has an associated *Contact List*. The frequency and traffic between two nodes can be used to first define the degree-of-familiarity between two nodes, then this information added to the node's *Contact List*. The *Contact List* can then be sorted according to familiarity degree. In this manner, we can track changes in the relationships between nodes and their contacts. Table 16.4 shows an example of tracking familiarity over time.

16.6. Communications Analysis

16.6.1. *First Communications Data Set*

In order to characterize the social network of interest, ideally we would like data to be made available for a reference population, in order to compare and contrast the behaviour(s) of our group with 'standard' communications patterns; to couch this in immunity-based computing (IBC) terminology: *self* versus *non-self*.[60] For instance, a typical person will tend to make regular calls to friends, family, and businesses (and most likely with this order of frequency). They will also typically opt for economical telephone services (be that via a plan or using a pre-paid service). Lastly, they will both expect and deliver quick responses (witness the rapid turnaround times with which teenagers text each other).

Given the above, any investigation of the data will necessarily be looking for anomalous patterns over different time frames — in other words, daily/weekly/seasonal (and similar) variations.

A preliminary static analysis of this first telephone call traffic data set revealed a high proportion of short-duration (<30 second) calls. Next, the ten most frequent callers during a 6-month period were identified. The top four of these are shown in Figure 16.11, with traffic split into bi-directional links where known.

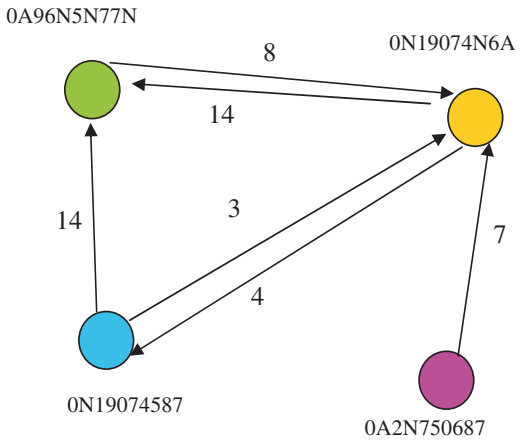

Fig. 16.11. Link Diagram (January–September 2004).

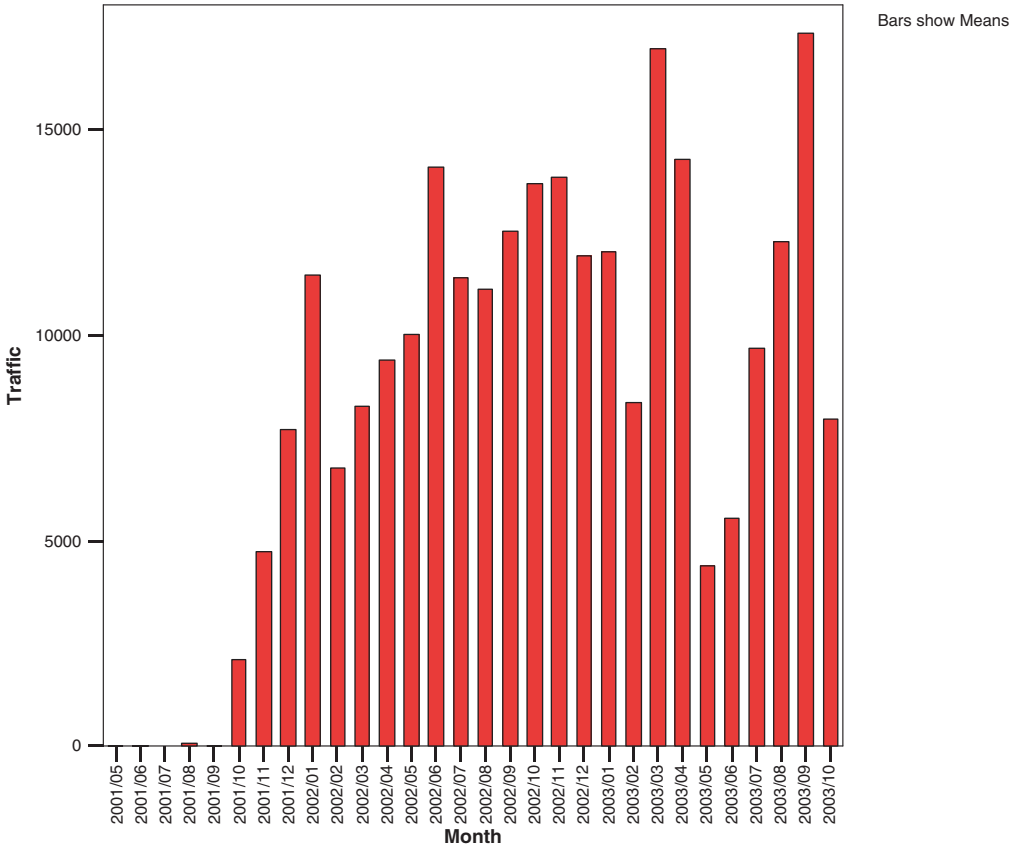

Fig. 16.12. Communications Traffic (24-month period).

Dynamic data analysis is of more interest than static however. In keeping with our own admonition *viz-a-viz* to visualize the data prior to invoking DM techniques in earnest, we simply plot the total traffic over the 2-year period, looking for general patterns of activity/inactivity. Figure 16.12 shows a gradual/steady increase during the first 12–14 months, with peaks at January and June of the first year, as well as March and September of the second year, together with a trough at May of

Table 16.4. Tracking Familiarity Over Time.

January 2000		February 2000	
Contact	Familiarity	Contact	Familiarity
A	0.57	A	0.64
B	0.23	B	0.20
C	0.15	C	0.16
D	0.05	D	0.10

Using this method, we can expect to discover associations between the contacts of a particular node. In addition, we can incorporate some period selection processes before conducting the aforementioned AR analysis. In this manner, we are able to determine contact associations during a nominated time interval.

16.5.4. *Familiarity Analysis*

As a second illustration of the usefulness of `NetShow 2.0`, let us consider the issue of 'familiarity'. As already observed, each network node has an associated *Contact List*. The frequency and traffic between two nodes can be used to first define the degree-of-familiarity between two nodes, then this information added to the node's *Contact List*. The *Contact List* can then be sorted according to familiarity degree. In this manner, we can track changes in the relationships between nodes and their contacts. Table 16.4 shows an example of tracking familiarity over time.

16.6. Communications Analysis

16.6.1. *First Communications Data Set*

In order to characterize the social network of interest, ideally we would like data to be made available for a reference population, in order to compare and contrast the behaviour(s) of our group with 'standard' communications patterns; to couch this in immunity-based computing (IBC) terminology: *self* versus *non-self*.[60] For instance, a typical person will tend to make regular calls to friends, family, and businesses (and most likely with this order of frequency). They will also typically opt for economical telephone services (be that via a plan or using a pre-paid service). Lastly, they will both expect and deliver quick responses (witness the rapid turnaround times with which teenagers text each other).

Given the above, any investigation of the data will necessarily be looking for anomalous patterns over different time frames — in other words, daily/weekly/seasonal (and similar) variations.

A preliminary static analysis of this first telephone call traffic data set revealed a high proportion of short-duration (<30 second) calls. Next, the ten most frequent callers during a 6-month period were identified. The top four of these are shown in Figure 16.11, with traffic split into bi-directional links where known.

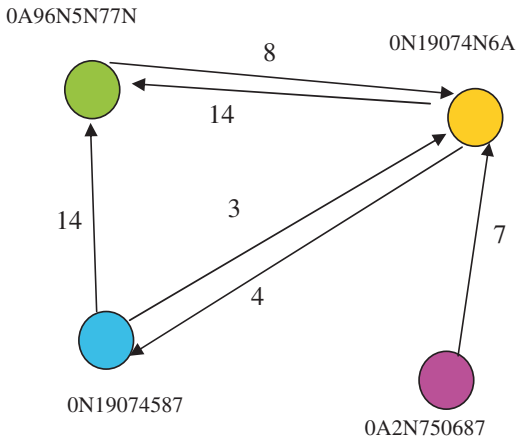

Fig. 16.11. Link Diagram (January–September 2004).

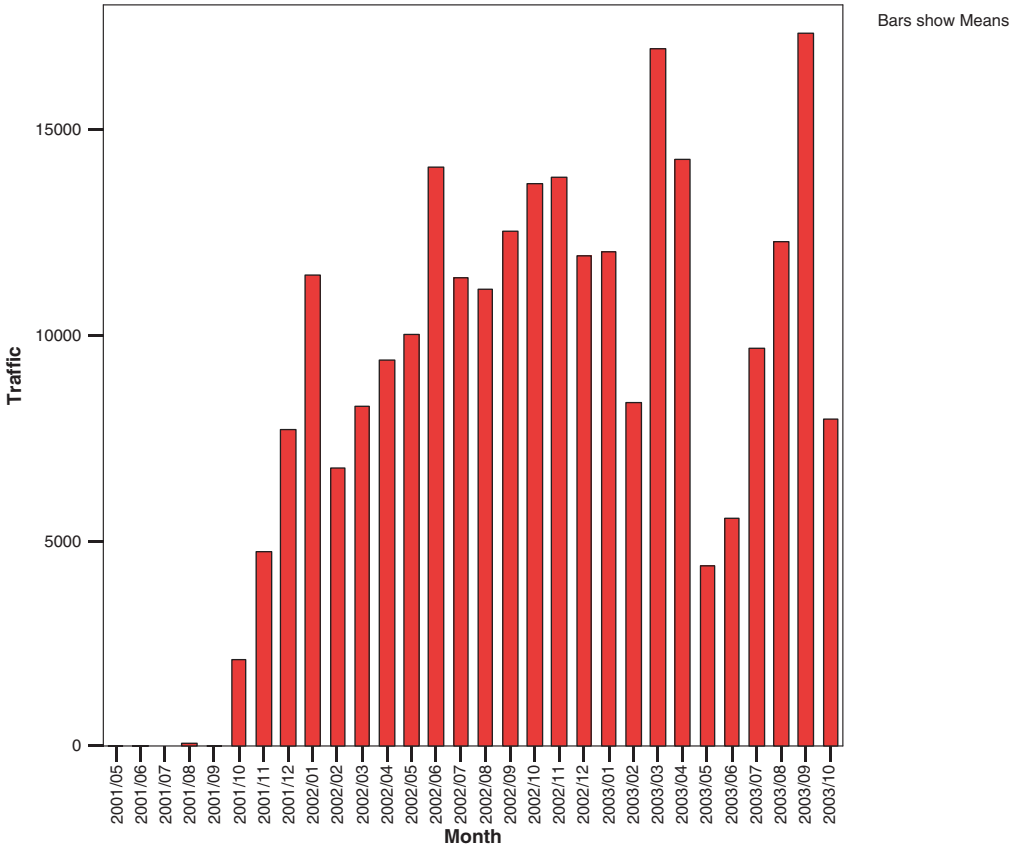

Fig. 16.12. Communications Traffic (24-month period).

Dynamic data analysis is of more interest than static however. In keeping with our own admonition *viz-a-viz* to visualize the data prior to invoking DM techniques in earnest, we simply plot the total traffic over the 2-year period, looking for general patterns of activity/inactivity. Figure 16.12 shows a gradual/steady increase during the first 12–14 months, with peaks at January and June of the first year, as well as March and September of the second year, together with a trough at May of

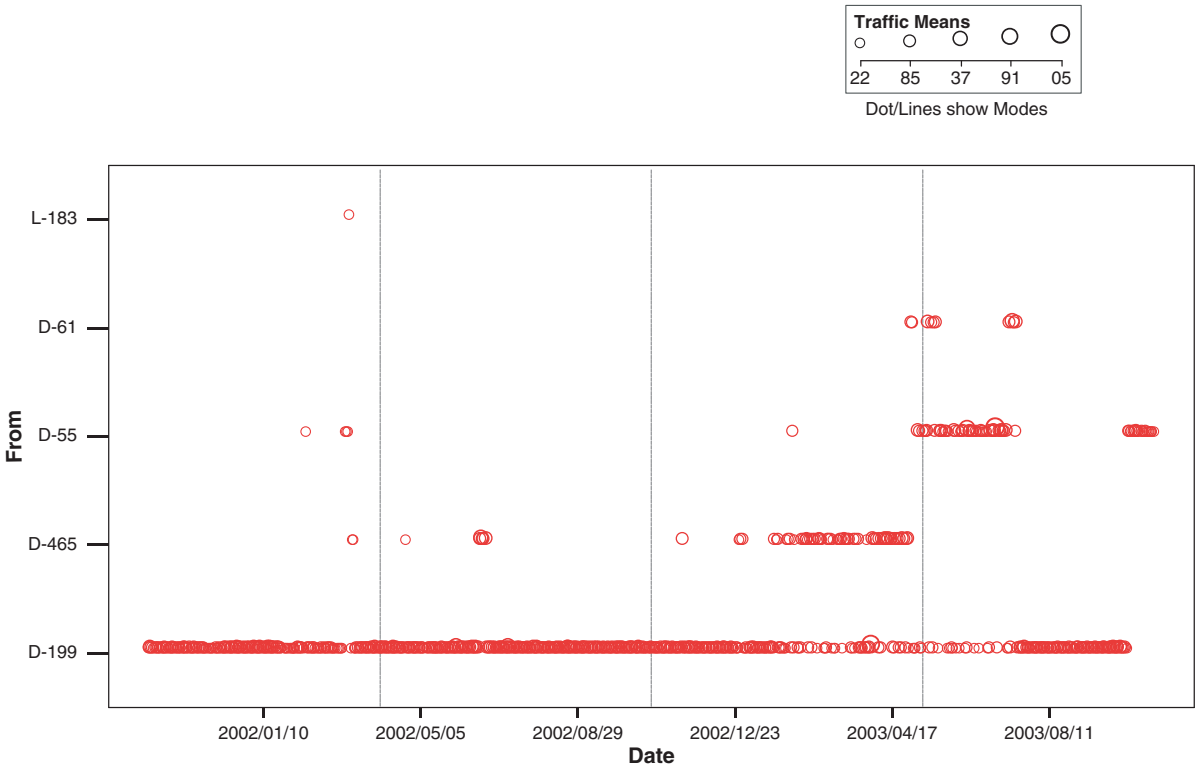

Fig. 16.13. Communications Traffic (13-month period).

the second year. Figure 16.13 shows traffic call patterns for five nominated nodes (services) during a shorter time frame.

Changes in network architecture reflect changes in the composition of the social network over time; likewise changes in link strengths reflect variations in inter-group communications. Secondly, correlations between callers may reveal useful information. For instance, do certain external events trigger call activity in other network nodes?

Four of the ten most frequent callers from March through August 2004 were plotted over the same time period, using an SPSS smoothing function[j] for ease of readability (Figure 16.14). It is obvious that not only are these data correlated, but that a call (trigger) sequence occurred, from node 19NN74N719 → 0586058N19 → 0586058N74 → 0NN7NA11NN.

One particular correlation was examined further, using the in-built SPSS cross-correlation tool. Figure 16.15 shows the correlation between nodes 0586058N19 and 0586058N74 during the period January–September 2004. A correlation coefficient of 40% is evident (Figure 16.16), however beyond ±2–3 days this value becomes somewhat meaningless. This is an unexpectedly strong correlation, and reveals the fact that the signal-to-noise ratio of the data is quite high.

An even more interesting result is that on several occasions strong correlations are observed for nodes with no *direct* connection between them. In other words,

[j]SPSS uses the T4253H smoothing function to create new series values based on a compound data smoother.[61]

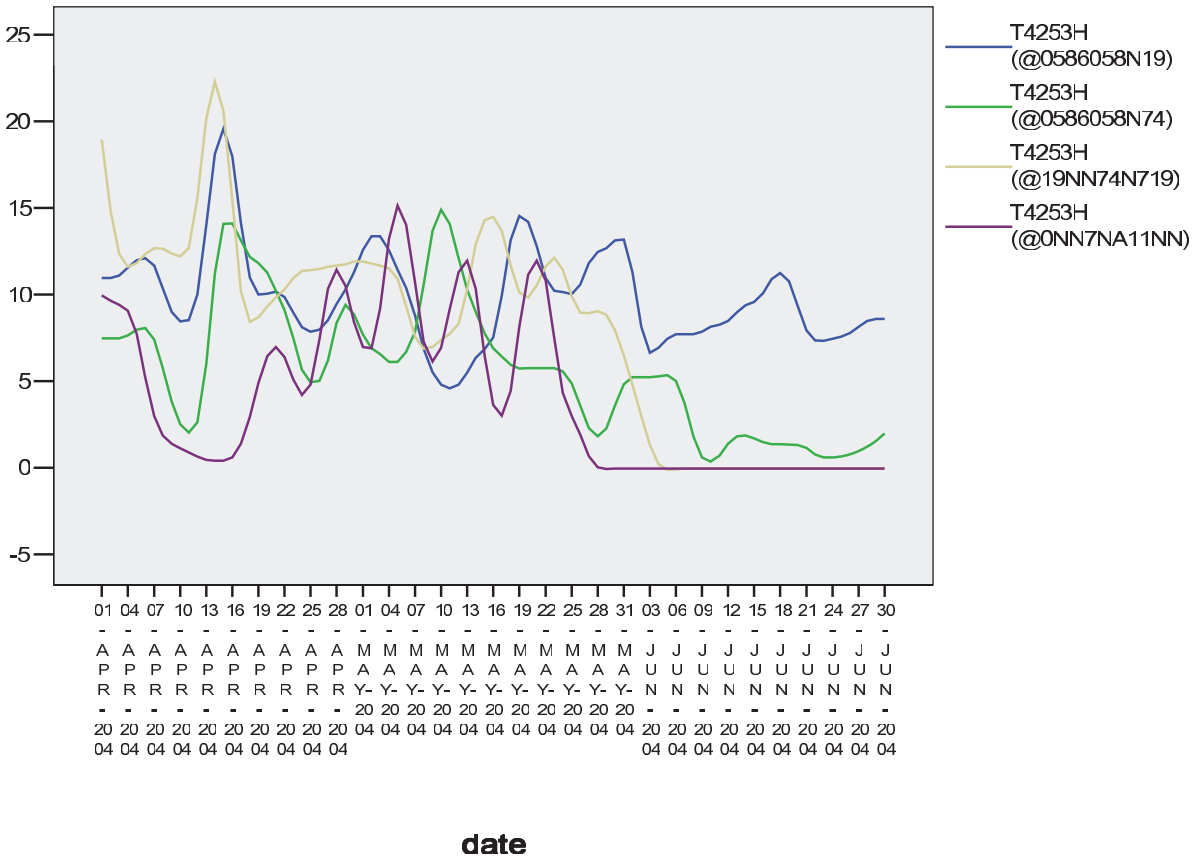

Fig. 16.14. The Top 4 Callers (April–June 2004).

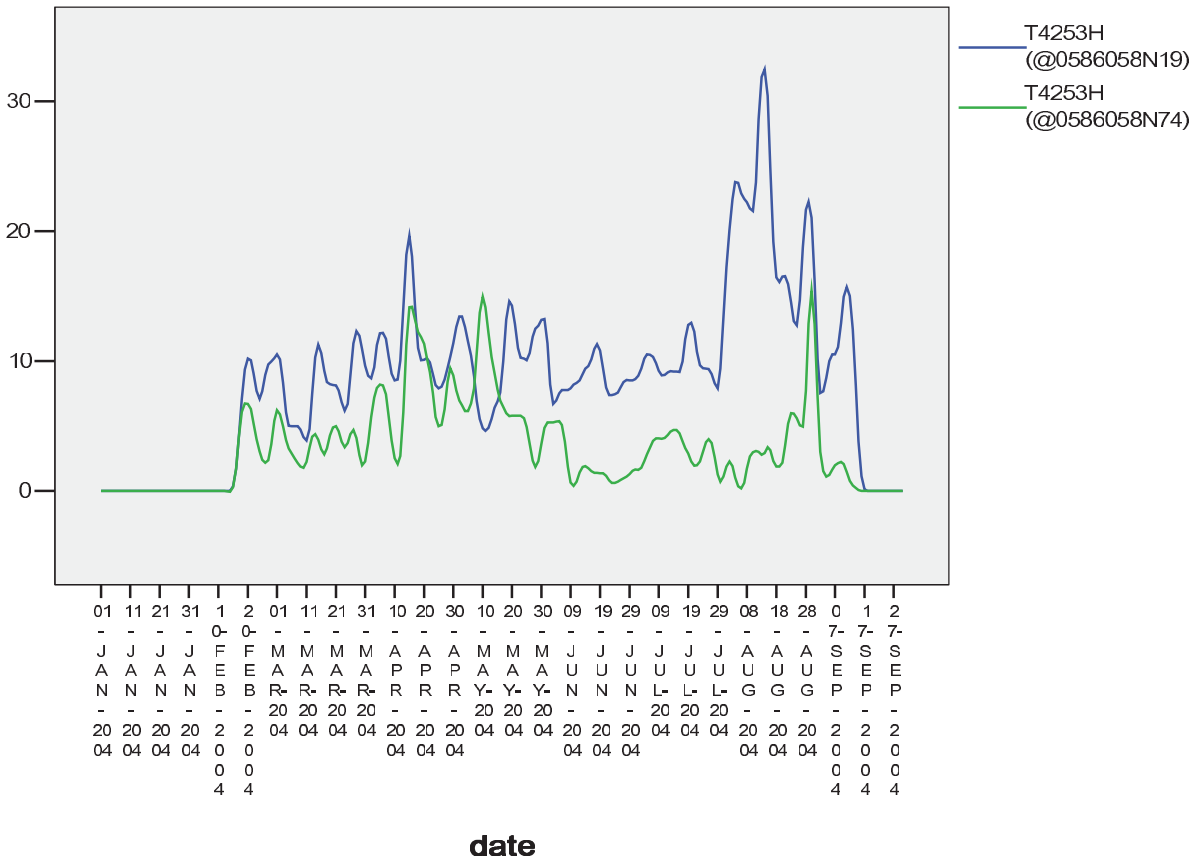

Fig. 16.15. 0586058N19 and 0586058N74 (January–September 2004).

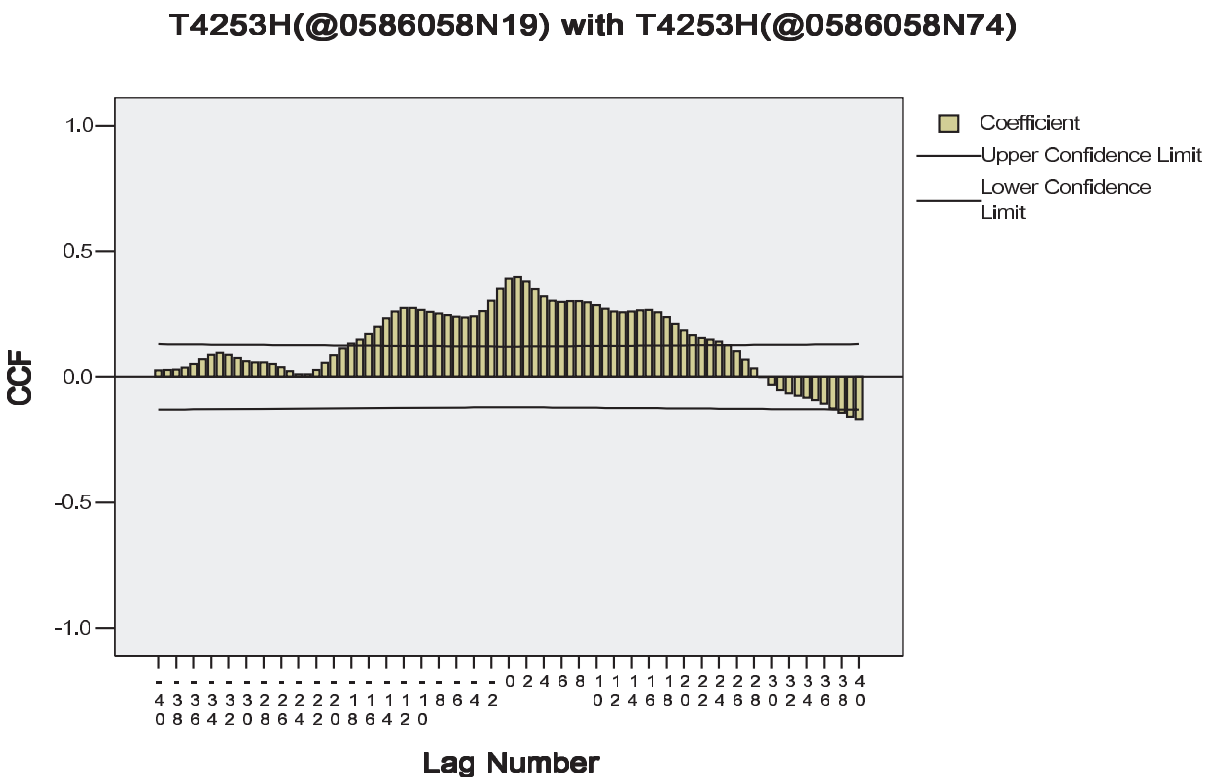

Fig. 16.16. 0586058N19 and 0586058N74 (January–September 2004).

indirect links (via a third-party common to both nodes) exist within this particular social network.

16.6.2. *Second Communications Data Set*

First and foremost, we identified five kinds of inbound service and nine kinds of outbound service (Tables 16.5 and 16.6, respectively) in this larger data set (258,000 records as opposed to 50,000 previously).

Two types of telephone service (D and L) dominate, accounting for over 99% of the *From* traffic, and 93% of the *To* traffic. We further note that just a couple of services account for the majority of telephone call traffic. More specifically, the number one Caller (D-61) is also the top callee; the number 3 caller (D-199) is the number 2 callee, and so on (Tables 16.7 and 16.8).

Thirdly, a large proportion of the call traffic takes place between the same type of service (D-D 53%; L-L 32%; D-L 6.3%; L-D 4%; D-I 1.4%; D-SN 0.62%, etc.).

Table 16.5. Inbound services.

Service Type	Number of IDs
D	4,273
L	2,046
OP	19
Y	14
I	3

Table 16.6. Outbound services.

Service Type	Number of IDs
D	5,026
L	2,845
L2	152
SN	152
OP	114
L1	84
SH	33
I	9
N	8

Table 16.7. The Top 10 Callers.

Rank	ID	Traffic	Rank in Caller List
1	D-61	23,704	1
2	D-465	14,283	7
3	D-199	10,686	2
4	D-55	7,325	4
5	L-183	6,785	3
6	L-13	6,285	11
7	D-181	5,967	10
8	L-973	5,702	28
9	D-1799	4,849	9
10	D-114	4,307	19

Table 16.8. The Top 10 Callers.

Rank	ID	Traffic	Rank in Caller List
1	D-61	19,511	1
2	D-199	10,378	3
3	L-183	7,916	5
4	D-55	7,836	4
5	L-1187	6,793	20
6	L-185	5,874	15
7	D-465	4,484	2
8	L-313	5,251	8
9	D-1799	4,230	9
10	D-181	4,160	7

A cross-correlation analysis of high-frequency callers did not yield any significant results, even invoking a 2-3 day time lag either side of zero. Nevertheless, as Figure 16.17 shows, reasonably high correlations are observed during certain time periods.

We experimented with manual determination of an 'optimum' time resolution (12, 6, 3, 1 month, 2 weeks, 1 week), and found that 1 month was preferable to 3 (since the latter tended to average out to a 'false' correlation level). We automated this process by way of a scripting (pre-processor) program which (i) automatically

Table 16.9. The Top 20 *From* Accounts.

Rank order	From Account	#Transactions	Rank order in *To* list
1	E-15	9	—
2	E-6	8	—
3	E-69*	7	A
4	E-12	6	—
5	E-24	6	—
6	E-14*	5	C
7	E-2	4	—
8	E-57	4	—
9	E-61	4	—
10	E-68	3	—
11	E-77	3	—
12	E-19*	2	S
13	E-34	2	—
14	E-39	2	—
15	E-43	2	—
16	E-50	2	—
17	E-71	2	—
18	E-87	2	—
19	E-92	2	—
20	E-10	1	—

Table 16.10. The Top 20 *To* Accounts.

Rank order	To Account	#Transactions	Rank order in *From* list
A	E-69*	13	3
B	E-13	4	—
C	E-14*	4	6
D	E-58	4	—
E	E-40	3	—
F	E-78	3	C
G	E-1	2	—
H	E-29	2	—
I	E-35	2	—
J	E-5	2	—
K	E-53	2	—
L	E-59	2	—
M	E-72	2	—
N	E-93	2	—
O	E-96	2	—
P	E-11	1	—
Q	E-16	1	—
R	E-18	1	—
S	E-19*	1	12
T	E-21	1	—

detects the highest correlations, and (ii) identifies the period(s) during which these occurs. It should be noted that this will involve variable starting times (i.e. they will not necessarily coincide with the start of a month). Again, no results of significance were forthcoming.

Table 16.11. The Top 20 *From* — *To* Pairs.

From Account	*To* Account	#transactions
E-69	E-69	6
E-12	E-13	4
E-14	E-14	4
E-57	E-58	4
E-77	E-78	3
E-2	E-1	2
E-6	E-29	2
E-34	E-35	2
E-12	E-53	2
E-15	E-59	2
E-2	E-69	2
E-68	E69	2
E-87	E-69	2
E-71	E-72	2
E-92	E-93	2
E-10	E-11	1
E-15	E-16	1
E-17	E-18	1
E-19	E-19	1
E-20	E-21	1

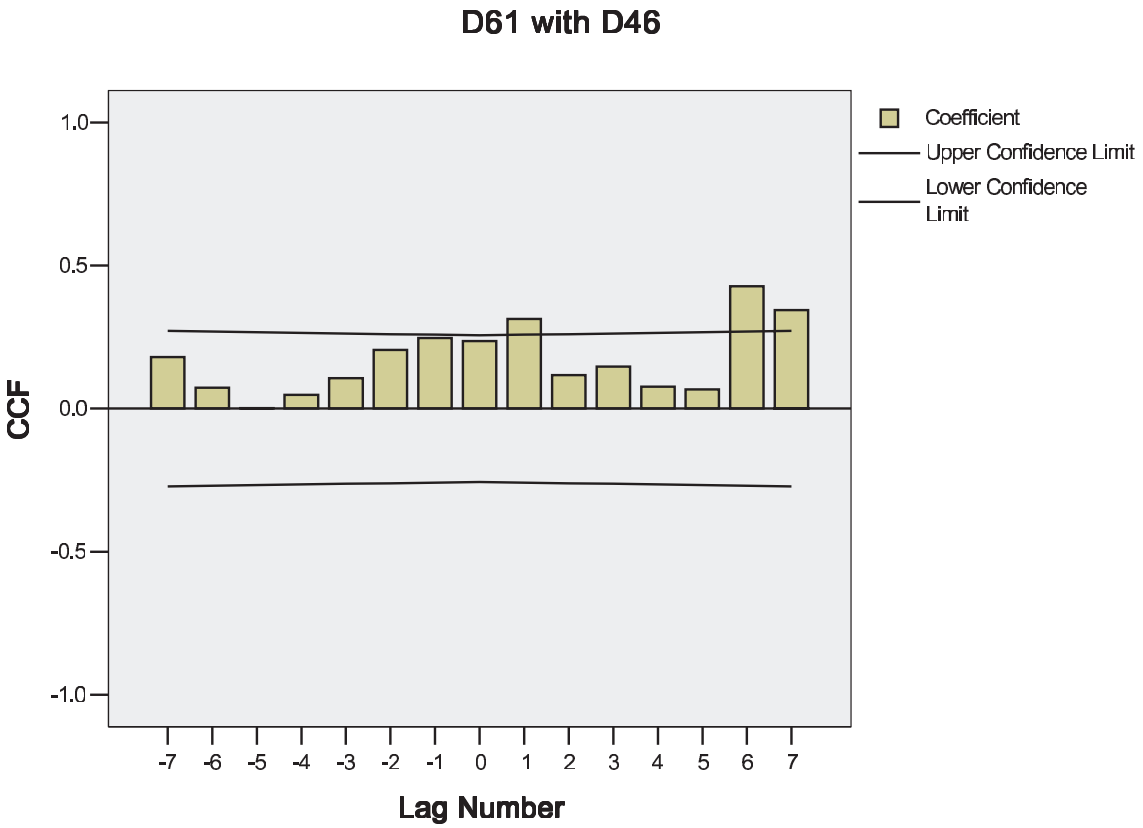

Fig. 16.17. D61-D46 Cross Correlation (1.3.03–30.4.03).

16.7. Network Dynamics

16.7.1. *Adding Meaningful Link and Path Weights to a Transaction Network*

As the current data-set is 'poor' in meaningful annotations, we first defined then added some meaningful weights to both network links and paths. The formula for calculating link/path weights is as follows:

Direct links (simplest case)

$$w_{ij} = \frac{t_{ij}\beta}{T_i} \quad \beta \leq 1; \;\; w_{ij} \leq 1 \tag{16.1}$$

where t_{ij} is the traffic from node-i to node-j, w_{ij} is the link weight from node-i to node-j, and T_i is the total (outgoing) traffic from node-i; β is a transaction type-related coefficient, and has different meanings for different kinds of transactions, and can be user-defined (Figure 16.18). The reason for adding this parameter is to denote the importance of a certain transaction. For example, in communication transactions, longer phone calls have higher β values.

Path weights

A *path* is defined as a route between two nodes. (Note that multiple paths can exist.)

Simple path weight

pw_{ik}, the link-weight from node-i to node-k (Figure 16.20) is given by:

$$\begin{aligned} pw_{14} &= w_{12}w_{24} \\ &= \left(\frac{t_{13}}{T_1}\right)\left(\frac{t_{24}}{T_2}\right) \\ &= \left(\frac{t_{13}}{t_{12}+t_{13}}\right)\left(\frac{t_{24}}{t_{24}+t_{25}}\right). \end{aligned} \tag{16.2}$$

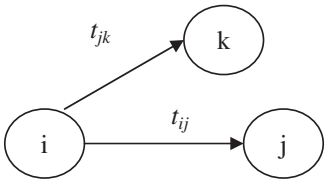

Fig. 16.18. Link/Path Weights.

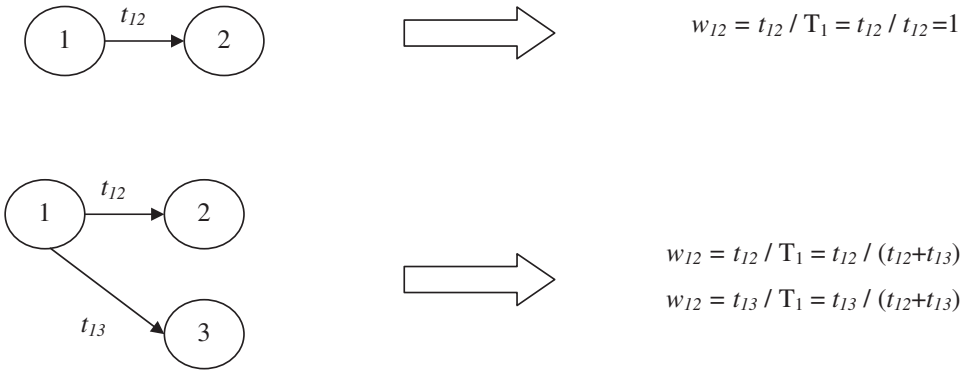

Fig. 16.19. Example Link/Path Weights.

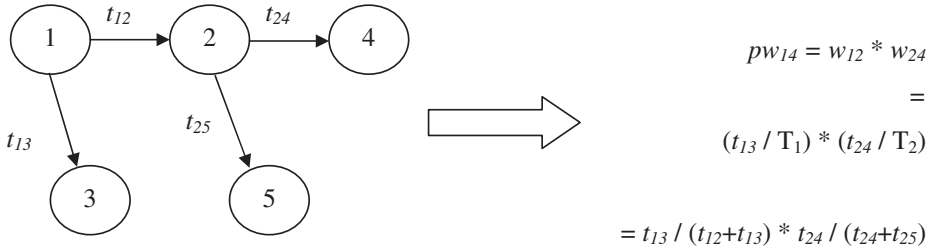

Fig. 16.20. Single Path.

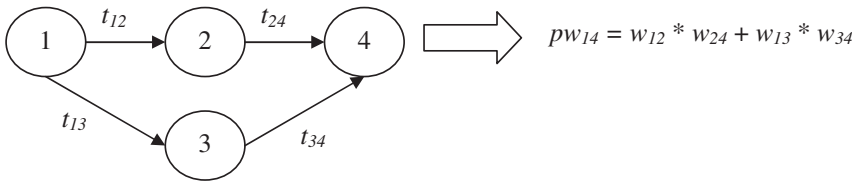

Fig. 16.21. Multiple Paths.

Multiple paths between two nodes

If there is more than one path between two nodes (Figure 16.21), the path weight between these two nodes is the sum of all individual path weights:

$$pw_{14} = w_{12}w_{24} + w_{13}w_{34} \tag{16.3}$$

Path weights for paths with overlapping links

If paths between two nodes have overlapping link(s), the link weight of the overlapped link is divided by the overlapped time when calculating the path weight. In the example of Figure 16.22, there are three paths between node-1 and node-4, with all three paths sharing a common link — l_{12} (the link between node-1 and node-2). Hence, in calculating pw_{14}, w_{12} needs to be divided by 3.

16.7. Network Dynamics

16.7.1. *Adding Meaningful Link and Path Weights to a Transaction Network*

As the current data-set is 'poor' in meaningful annotations, we first defined then added some meaningful weights to both network links and paths. The formula for calculating link/path weights is as follows:

Direct links (simplest case)

$$w_{ij} = \frac{t_{ij}\beta}{T_i} \quad \beta \leq 1; \;\; w_{ij} \leq 1 \tag{16.1}$$

where t_{ij} is the traffic from node-i to node-j, w_{ij} is the link weight from node-i to node-j, and T_i is the total (outgoing) traffic from node-i; β is a transaction type-related coefficient, and has different meanings for different kinds of transactions, and can be user-defined (Figure 16.18). The reason for adding this parameter is to denote the importance of a certain transaction. For example, in communication transactions, longer phone calls have higher β values.

Path weights

A *path* is defined as a route between two nodes. (Note that multiple paths can exist.)

Simple path weight

pw_{ik}, the link-weight from node-i to node-k (Figure 16.20) is given by:

$$\begin{aligned} pw_{14} &= w_{12}w_{24} \\ &= \left(\frac{t_{13}}{T_1}\right)\left(\frac{t_{24}}{T_2}\right) \\ &= \left(\frac{t_{13}}{t_{12}+t_{13}}\right)\left(\frac{t_{24}}{t_{24}+t_{25}}\right). \end{aligned} \tag{16.2}$$

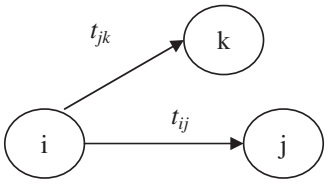

Fig. 16.18. Link/Path Weights.

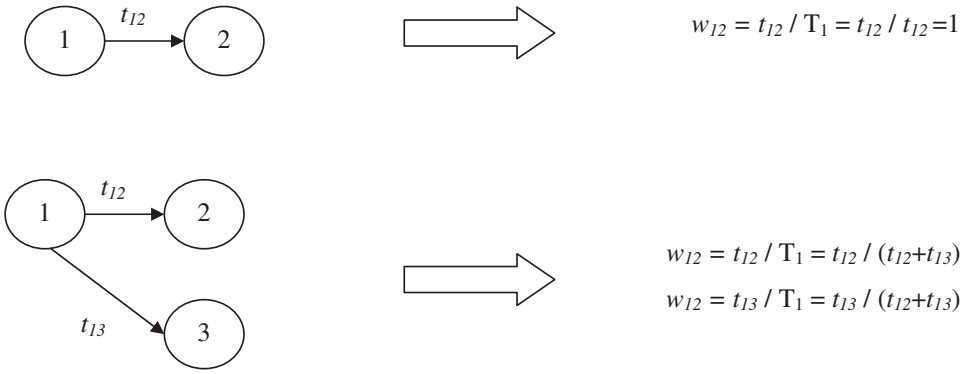

Fig. 16.19. Example Link/Path Weights.

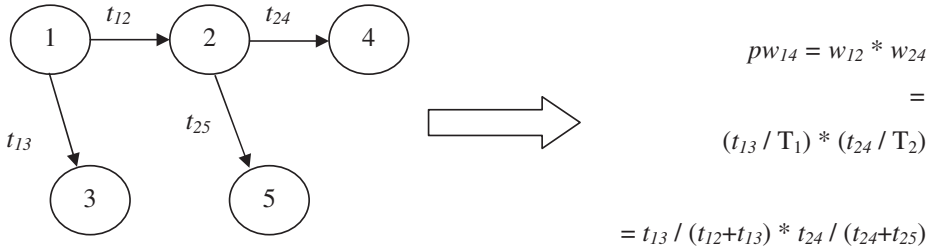

Fig. 16.20. Single Path.

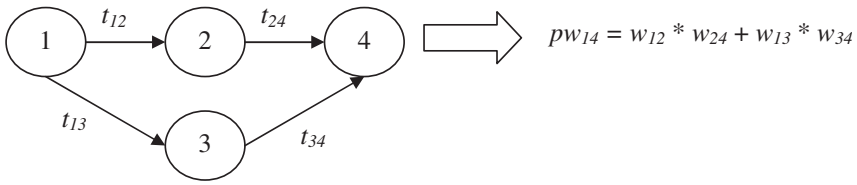

Fig. 16.21. Multiple Paths.

Multiple paths between two nodes

If there is more than one path between two nodes (Figure 16.21), the path weight between these two nodes is the sum of all individual path weights:

$$pw_{14} = w_{12}w_{24} + w_{13}w_{34} \tag{16.3}$$

Path weights for paths with overlapping links

If paths between two nodes have overlapping link(s), the link weight of the overlapped link is divided by the overlapped time when calculating the path weight. In the example of Figure 16.22, there are three paths between node-1 and node-4, with all three paths sharing a common link — l_{12} (the link between node-1 and node-2). Hence, in calculating pw_{14}, w_{12} needs to be divided by 3.

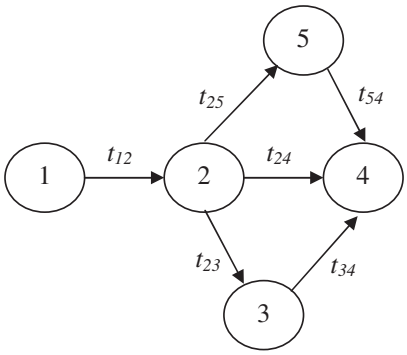

Fig. 16.22. Overlapping Paths.

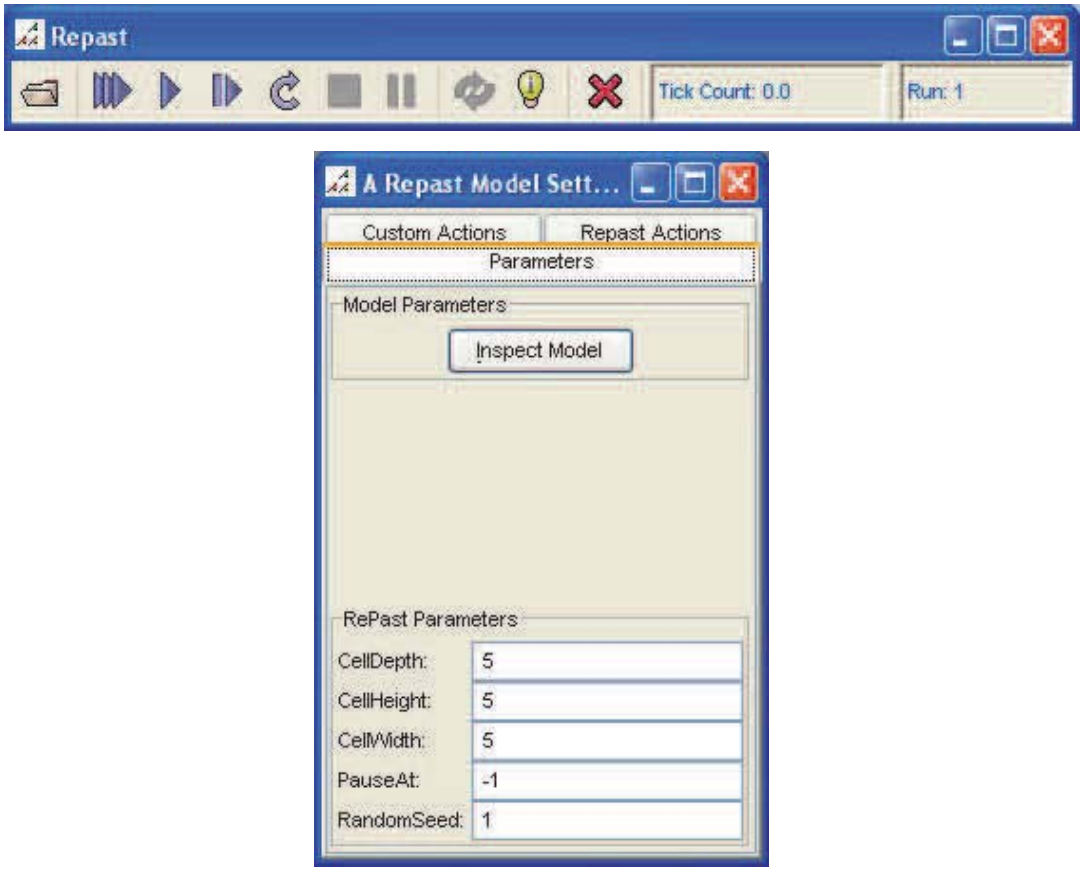

Fig. 16.23. Main SWARM User Interface.

16.7.2. *Building SWARM Simulations to Display Network Dynamics*

The main purpose of incorporating SWARM techniques in this project is twofold: (i) to demonstrate (model) dynamic network changes, and (ii) to (potentially) enable network prediction. The tool currently being used in this endeavour is `Repast`. We have performed some preliminary work to embed a SWARM simulation into our system. Figure 16.23 shows the SWARM Graphical User Interface, whereas

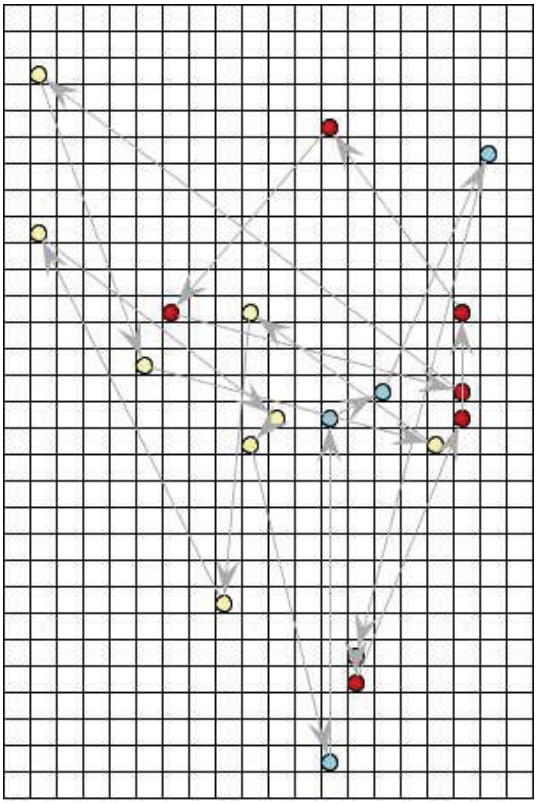

Fig. 16.24. 2D Network View.

Figures 16.24 through 16.26 are snapshots of typical (dynamic) 2-Dimensional and 3-Dimensional network views.

Each agent (node) in the output window represents a person in the network. Node positions show the relationships between people in the network; the more contacts that people make, the closer the relationship between them. The time tick of the output update can be daily based. The input of the simulation is transaction data sets. Accordingly, the simulation can show not only the network life-cycle, but also how relationships among members of the network change over time.

16.8. Public Domain Data

In Sect. 16.5, we stated that a long-term aim with the present project was to model social networks with sufficient accuracy to enable the 'reverse-engineering' of time series data leading up to known real-world events (given the availability of well documented case studies, that is). The data sets at our disposal however can be best characterized as 'data rich, knowledge poor', as previously observed. The question arises then as to whether more attribute-rich data sets are available, and more especially available in the public domain (which would overcome the need for data anonymity). The short answer is that such data sets *do* exist, one example being `Tweed`.[62,63k] However we should point out that this latter data set differs markedly

[k]http://www.uib.no/people/sspje/tweed.htm

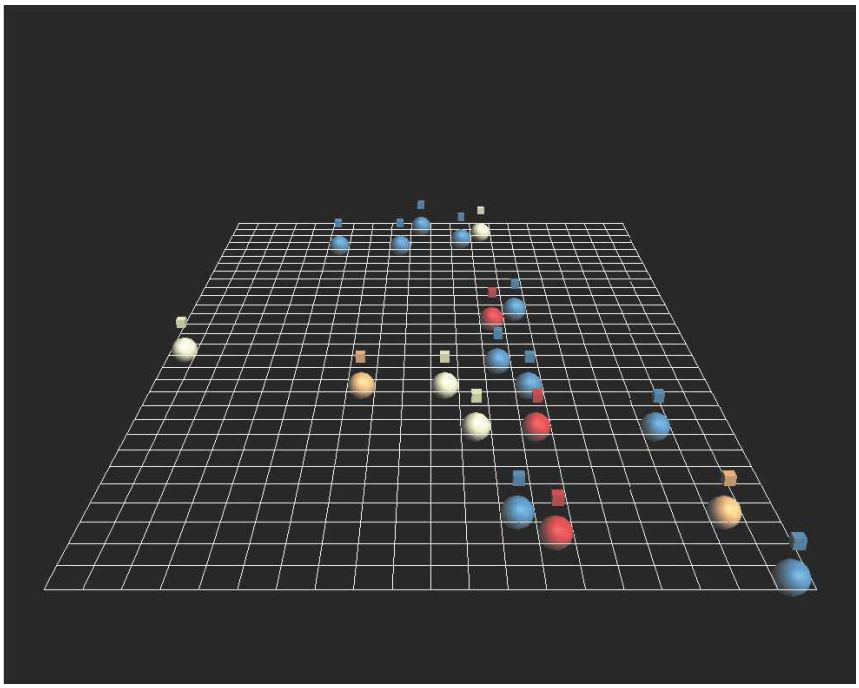

Fig. 16.25. 3D Network View (nodes only).

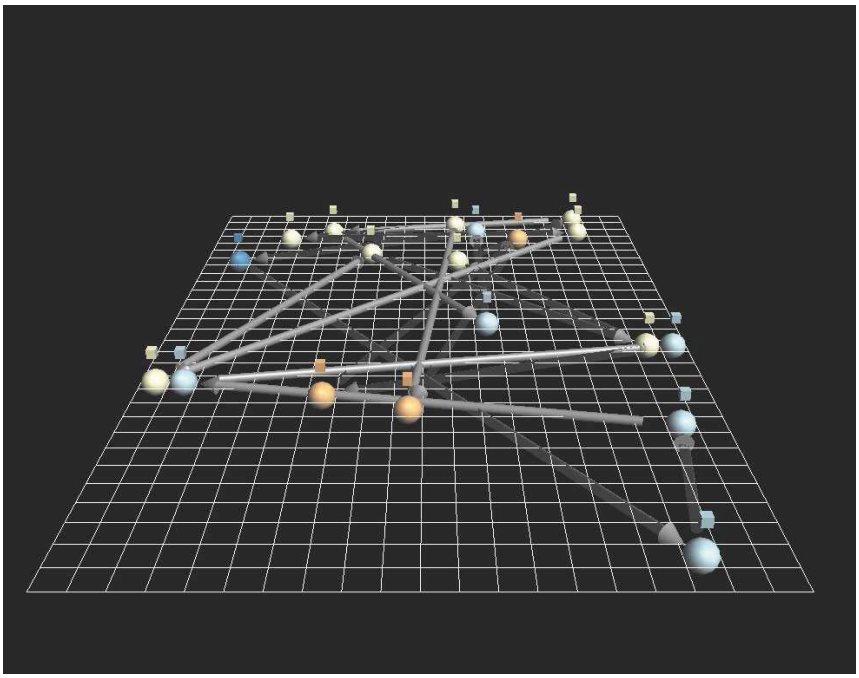

Fig. 16.26. 3D Network View (nodes and links).

from the communications traffic data sets considered up to now. First and foremost, `Tweed` records many *different* social networks, not just the one (as with the present study). Secondly, the records contain case summaries ('macro' data, if you will), rather than 'micro' data.

16.9. Conclusion and Suggestions for Further Work

We have devised a model to enable the development of agent-based systems for modelling (and by extension, prediction), and which can incorporate not only agents, but also neural network, evolutionary algorithm, swarm, machine learning (e.g. pattern recognition), data mining/link mining, and statistical techniques.

There are several promising avenues to explore in future studies, in order to build upon the foundations we have laid with `NetShow 2.0`, these being:

(1) validation of our agent-based model (ABM) and system, beyond just the (limited) data sets available during the course of the present study,
(2) adaptation of `NetShow 2.0` in order to handle other data sets (e.g. `Tweed`),
(3) correlation of data with 'events', in order to analyze patterns leading up to such events, with a view to predicting 'similar' such occurrences in the future,
(4) combining agent-based learning (= a local technique) with Data/Link Mining (= a global technique), in order to discover more about relationships between nodes (accounts, people and phone owners), changing trends, and other previously undiscovered information/knowledge,
(5) comparing system- *versus* local-level agent learning, combining agents with link mining in the former, and agents and pattern matching for the latter,
(6) combining agent-based learning with Computational Intelligence (e.g. ANNs, evolutionary algorithms, swarms) and/or Machine Learning techniques, to enhance knowledge discovery from network data,
(7) integration of telephony and geographical location data (although we appreciate the inherent problems in adequately de-referencing/aliasing the latter),
(8) further research into how different data types can be combined, namely nominal, ordinal, interval and ratio (the data sets we have been working with up to now have been almost exclusively nominal),
(9) development of alternative data visualization/presentation formats/views (e.g. nodes being denoted by different colours and/or shapes),
(10) further development of swarm-based visualization of network dynamics (in contrast to `NetShow 2.0`, which is essentially restricted to static displays),
(11) further investigation into pattern discovery from *Contact Lists*,
(12) further development of *degree-of-familiarity* and the tracking of same over time, as well as other metrics (e.g. 'similarity').

Let us consider one of these recommendations in a little more detail (8 above). Both the telephone call traffic and financial transaction data sets involve nominal data. The question arises as to whether the techniques developed above apply equally well to data sets which incorporate other data types, such as ordinal, interval, and ratio data (again, this could be relevant to data sets such as `Tweed`). In other words, can objects (entities) described in terms of different data types be combined together? Fundamentally, this is not feasible unless there is some underlying *relationship* between the data in question (which could, for example be time).

One potential approach is to use fuzzy reasoning combined with agent-based leaning.[64,65] This would involve the following four steps:

(1) Define four *fuzzy variables* to describe each data group, say N, O, I, and R,
(2) Build *fuzzy membership functions* for each fuzzy variable, based on the features of each individual data group.
(3) Derive *fuzzy rules* based on the problem domain, in which relationships exist between the four data groups.
(4) Build *reasoning mechanisms* to evaluate these data by use of *agent learning* and *reasoning*, assuming the measurements can be related to different purposes. Supposing we choose a BDI agent to implement the data evaluation, then

- Fuzzy rules correspond to *belief*,
- Measurement guidelines, such as 'trend of changes', correspond to *intention*, and
- Different views or actions relating to multiple measurement purposes can be modelled as *plan*.

For subsequent data analysis, we might use four variables to describe an object in 4-dimensional space, then analyze the projection of the object onto lower-dimension space, in order to classify or cluster these objects by applying Data Mining techniques based on the requirements for data analysis. (Note that the above proposed method is predicated upon *relationships* actually existing between these four data types.)

Acknowledgment

The authors would like to acknowledge the support of the Intelligent Systems Research Centre at the University of Wollongong.

References

1. C. Westphal and T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*. (Wiley, New York, NY, 1998).
2. I. Whitten and E. Frank, *Practical Data Mining: Machine Learning Tools and Techniques*. (Morgan Kaufmann, San Francisco, CA, 2005).
3. J. Fulcher. Computational Intelligence: An Introduction. In eds. J. Fulcher and L. Jain, *Computational Intelligence: a Compendium*. Springer-Verlag, Berlin, Germany, (2008).
4. J. Fulcher and L. Jain, Eds., *Computational Intelligence: A Compendium*. (Springer-Verlag, Berlin, Germany, 2008).
5. L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*. (Wiley, New York, NY, 2004).
6. M. Wooldridge, *An Introduction to Multiagent Systems*. (Wiley, Chichester, UK, 2002).
7. M. Zhang, B. Bai, F. Ren and J. Fulcher. Collaborative Agents for Complex Problem Solving. In eds. C. Mumford and L. Jain, *Collaborative Computational Intelligence*. Springer-Verlag, Berlin, Germany, (2008).

8. D. Clouse, C. Giles, B. Horne and G. Cottrell, Time delay neural networks: representation and induction of finite-state machines, *IEEE Trans. Neural Networks.* **8**, 1065–1070, (1997).
9. M. Zhang, Ed., *Artificial Higher-Order Neural Networks for Economics and Business.* (Idea Group International, Hershey, PA, 2008).
10. S. Abe, Ed., *Support Vector Machines for Pattern Classification.* (Springer-Verlag, New York, NY, 2005).
11. J. Shawe-Taylor and N. Christianni, Eds., *Support Vector Machines and Other Kernel Based Learning Methods.* (Cambridge University Press, Cambridge, UK, 2000).
12. T. Back, *Evolutionary Algorithms in Theory and Practice.* (Oxford University Press, New York, NY, 1996).
13. L. Davis, Ed., *Handbook of Genetic Algorithms.* (Van Nostrand Reinhold, New York, NY, 1991).
14. D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* (Wiley, New York, NY, 1999).
15. D. Goldberg and K. Deb, *Foundations of Genetic Algorithms: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms.* (Morgan Kaufmann, San Mateo, CA, 1991).
16. J. Holland, *Adaptation in Natural and Artificial Systems (2nd ed.).* (MIT Press, Cambridge, MA, 1992).
17. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* (Springer-Verlag, Berlin, Germany, 1996).
18. M. Mitchell, *Introduction to Genetic Algorithms.* (MIT Press, Cambridge, MA, 1995).
19. W. Banzhaf, *Genetic Programming: An Introduction.* (Morgan Kaufmann, San Francisco, CA, 1998).
20. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* (MIT Press, Cambridge, MA, 1992).
21. W. Langdon, *Data Structures and Genetic Programming: GP + Data Structures = Automatic Programming.* (Kluwer Academic Publishers, Boston, MA, 1998).
22. W. Langdon, R. Poli, N. McPhee and J. Koza. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In eds. J. Fulcher and L. Jain, *Computational Intelligence: a Compendium.* Springer-Verlag, Berlin, Germany, (2008).
23. V. Buskens, *Social Networks and Trust.* (Kluwer, Boston, MA, 2002).
24. L. Freeman, *The Development of Social Network Analysis: A Study in the Sociology of Science.* (Empirical Press, Vancouver, Canada, 2004).
25. J. Scott, *Social Network Analysis: A Handbook (2nd ed.).* (Sage, Newberry Park, CA, 2000).
26. S. Wasserman and K. Faust, Eds., *Social Networks Analysis: Methods and Applications.* (Cambridge University Press, Cambridge, UK, 1994).
27. C. Date, *Introduction to Data Base Systems.* (Addison Wesley, Reading, MA, 2004).
28. D. Kroenke, *Data Base Processing: Fundamentals Design and Implementation.* (Prentice Hall, Englewood Cliffs, NJ, 2004).
29. P. Rob, *Data Base Systems: Design, Implementation, and Management.* (Course Technology, Cambridge, MA, 2004).
30. M. Huisman and M. Van Duijn. Software for Social Network Analysis. In eds. P. J. Carrington, J. Scott and S. Wasserman, *Models and Methods in Social Network Analysis*, pp. 270–316. Cambridge University Press, New York, NY, (2005).
31. W. Nooy, A. Mrvar and V. Batageli, Eds., *Exploratory Social Network Analysis with Pajek.* (Cambridge University Press, Cambridge, UK, 2005).

One potential approach is to use fuzzy reasoning combined with agent-based leaning.[64,65] This would involve the following four steps:

(1) Define four *fuzzy variables* to describe each data group, say N, O, I, and R,
(2) Build *fuzzy membership functions* for each fuzzy variable, based on the features of each individual data group.
(3) Derive *fuzzy rules* based on the problem domain, in which relationships exist between the four data groups.
(4) Build *reasoning mechanisms* to evaluate these data by use of *agent learning* and *reasoning*, assuming the measurements can be related to different purposes. Supposing we choose a BDI agent to implement the data evaluation, then

- Fuzzy rules correspond to *belief*,
- Measurement guidelines, such as 'trend of changes', correspond to *intention*, and
- Different views or actions relating to multiple measurement purposes can be modelled as *plan*.

For subsequent data analysis, we might use four variables to describe an object in 4-dimensional space, then analyze the projection of the object onto lower-dimension space, in order to classify or cluster these objects by applying Data Mining techniques based on the requirements for data analysis. (Note that the above proposed method is predicated upon *relationships* actually existing between these four data types.)

Acknowledgment

The authors would like to acknowledge the support of the Intelligent Systems Research Centre at the University of Wollongong.

References

1. C. Westphal and T. Blaxton, *Data Mining Solutions: Methods and Tools for Solving Real-World Problems*. (Wiley, New York, NY, 1998).
2. I. Whitten and E. Frank, *Practical Data Mining: Machine Learning Tools and Techniques*. (Morgan Kaufmann, San Francisco, CA, 2005).
3. J. Fulcher. Computational Intelligence: An Introduction. In eds. J. Fulcher and L. Jain, *Computational Intelligence: a Compendium*. Springer-Verlag, Berlin, Germany, (2008).
4. J. Fulcher and L. Jain, Eds., *Computational Intelligence: A Compendium*. (Springer-Verlag, Berlin, Germany, 2008).
5. L. Padgham and M. Winikoff, *Developing Intelligent Agent Systems: A Practical Guide*. (Wiley, New York, NY, 2004).
6. M. Wooldridge, *An Introduction to Multiagent Systems*. (Wiley, Chichester, UK, 2002).
7. M. Zhang, B. Bai, F. Ren and J. Fulcher. Collaborative Agents for Complex Problem Solving. In eds. C. Mumford and L. Jain, *Collaborative Computational Intelligence*. Springer-Verlag, Berlin, Germany, (2008).

8. D. Clouse, C. Giles, B. Horne and G. Cottrell, Time delay neural networks: representation and induction of finite-state machines, *IEEE Trans. Neural Networks.* **8**, 1065–1070, (1997).
9. M. Zhang, Ed., *Artificial Higher-Order Neural Networks for Economics and Business.* (Idea Group International, Hershey, PA, 2008).
10. S. Abe, Ed., *Support Vector Machines for Pattern Classification.* (Springer-Verlag, New York, NY, 2005).
11. J. Shawe-Taylor and N. Christianni, Eds., *Support Vector Machines and Other Kernel Based Learning Methods.* (Cambridge University Press, Cambridge, UK, 2000).
12. T. Back, *Evolutionary Algorithms in Theory and Practice.* (Oxford University Press, New York, NY, 1996).
13. L. Davis, Ed., *Handbook of Genetic Algorithms.* (Van Nostrand Reinhold, New York, NY, 1991).
14. D. Fogel, *Evolutionary Computation: Toward a New Philosophy of Machine Intelligence.* (Wiley, New York, NY, 1999).
15. D. Goldberg and K. Deb, *Foundations of Genetic Algorithms: A Comparative Analysis of Selection Schemes Used in Genetic Algorithms.* (Morgan Kaufmann, San Mateo, CA, 1991).
16. J. Holland, *Adaptation in Natural and Artificial Systems (2nd ed.).* (MIT Press, Cambridge, MA, 1992).
17. Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Programs.* (Springer-Verlag, Berlin, Germany, 1996).
18. M. Mitchell, *Introduction to Genetic Algorithms.* (MIT Press, Cambridge, MA, 1995).
19. W. Banzhaf, *Genetic Programming: An Introduction.* (Morgan Kaufmann, San Francisco, CA, 1998).
20. J. Koza, *Genetic Programming: On the Programming of Computers by Means of Natural Selection.* (MIT Press, Cambridge, MA, 1992).
21. W. Langdon, *Data Structures and Genetic Programming: GP + Data Structures = Automatic Programming.* (Kluwer Academic Publishers, Boston, MA, 1998).
22. W. Langdon, R. Poli, N. McPhee and J. Koza. Genetic Programming: An Introduction and Tutorial, with a Survey of Techniques and Applications. In eds. J. Fulcher and L. Jain, *Computational Intelligence: a Compendium.* Springer-Verlag, Berlin, Germany, (2008).
23. V. Buskens, *Social Networks and Trust.* (Kluwer, Boston, MA, 2002).
24. L. Freeman, *The Development of Social Network Analysis: A Study in the Sociology of Science.* (Empirical Press, Vancouver, Canada, 2004).
25. J. Scott, *Social Network Analysis: A Handbook (2nd ed.).* (Sage, Newberry Park, CA, 2000).
26. S. Wasserman and K. Faust, Eds., *Social Networks Analysis: Methods and Applications.* (Cambridge University Press, Cambridge, UK, 1994).
27. C. Date, *Introduction to Data Base Systems.* (Addison Wesley, Reading, MA, 2004).
28. D. Kroenke, *Data Base Processing: Fundamentals Design and Implementation.* (Prentice Hall, Englewood Cliffs, NJ, 2004).
29. P. Rob, *Data Base Systems: Design, Implementation, and Management.* (Course Technology, Cambridge, MA, 2004).
30. M. Huisman and M. Van Duijn. Software for Social Network Analysis. In eds. P. J. Carrington, J. Scott and S. Wasserman, *Models and Methods in Social Network Analysis*, pp. 270–316. Cambridge University Press, New York, NY, (2005).
31. W. Nooy, A. Mrvar and V. Batageli, Eds., *Exploratory Social Network Analysis with Pajek.* (Cambridge University Press, Cambridge, UK, 2005).

32. J. Mena, *Investigative Data Mining for Security and Criminal Detection*. (Butterworth-Heinemann, Boston, MA, 2003).
33. M. Sageman, *Understanding Terror Networks*. (University of Philadelphia Press, Philadelphia, PA, 2004).
34. D. Farley, Breaking al qaeda cells: a mathematical analysis of counterterrorism, *Operations: Studies in Conflict Terrorism*. **26**, 399–411, (2003).
35. N. Memon and H. Larsen. Practical approaches for analysis, visualization and destabilizing terrorist networks. In *Proc. 1st Intl. Conf. Availability, Reliability and Security (ARES'06)*, pp. 906–913, Los Alamitos, CA, (2006). IEEE Computer Society Press.
36. V. Latora and M. Marchiori, How the science of complex networks can help develop strategies against terrorism, *Chaos, Solitons and Fractals*. **20**, 69–75, (2004).
37. K. Pattipati, P. Willet, J. Allanach, H. Tu and S. Singh. Hidden Markov Models and Bayesian Networks for counter-terrorism. In eds. R. Popp and J. Yen, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, pp. 25–50. IEEE Press, Piscataway, NJ, (2006).
38. L. Rabiner and B. Juang, An introduction to hidden markov models, *IEEE ASSP Magazine*. **January**, 4–16, (1986).
39. F. Jensen, *Bayesian Networks and Decision Graphs*. (Springer-Verlag, Berlin, Germany, 2001).
40. K. Korb, *Bayesian Artificial Intelligence*. (CRC Press, Bocas Raton, FL, 2004).
41. R. Neapolitan, *Learning Bayesian Networks*. (Prentice Hall, Englewood Cliffs, NJ, 2003).
42. T. Mifflin, C. Boner, G. Godfrey and M. Greenblatt. Detecting terrorist activities in the twenty-first century: a theory of detection for transactional networks. In eds. R. Popp and J. Yen, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, pp. 349–365. IEEE Press, Piscataway, NJ, (2006).
43. D. Skillicorn. Social network analysis via matrix decomposition. In eds. R. Popp and J. Yen, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, pp. 367–391. IEEE Press, Piscataway, NJ, (2006).
44. J. Golbeck, A. Mannes and J. Hendler. Semantic web technologies for terrorist network analysis. In eds. R. Popp and J. Yen, *Emergent Information Technologies and Enabling Policies for Counter-Terrorism*, pp. 125–137. IEEE Press, Piscataway, NJ, (2006).
45. L. Getoor, Link mining: a new data mining challenge, *SIGKDD Explorations*. **4**, 1–6, (2003).
46. M. Sparrow, The application of network analysis to criminal intelligence: an assessment of the prospects, *Social Networks*. **13**, 251–274, (1991).
47. T. Fawcett and F. Provost. Combining data mining and machine learning for effective user profiling. In *Proc. 2nd Intl. Conf. on Knowledge Discovery and Data Mining (KDD'96)*, Portland, OR, 2.
48. T. Fawcett and F. Provost. Activity monitoring: Noticing interesting changes in behavior. In *Proc. 5th Intl. Conf. on Knowledge Discovery and Data Mining (KDD'99)*, San Diego, CA, 15.
49. T. Senator, H. Goldberg, *et al.* The financial crimes enforcement network ai system (fais), *AI Magazine*. **16**, 21–39, (1995).
50. T. Senator, Link mining applications: Progress and challenges, *SIGKDD Explorations*. **7**, 76–83, (2006).
51. J. Ferber, *Multi-Agent Systems: An Introduction to Distributed Artificial Intelligence*. (Addison Wesley, Reading, MA, 1999).
52. M. Wooldridge and N. Jennings, Intelligent agents: theory and practice, *The Knowledge Engineering Review*. **10**, 115–152, (1995).

53. A. Abraham, C. Grosan and V. Ramos, Eds., *Swarm Intelligence in Data Mining*. (Springer-Verlag, Berlin, Germany, 2006).
54. A. Engelbrecht, *Fundamentals of Computational Swarm Intelligence (2nd ed.)*. (Wiley, Hoboken, NJ, 2005).
55. F. Luna and B. Stefansson, Eds., *Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming*. (Kluwer Academic Publishers, Dordrecht, The Netherlands, 2000).
56. P. Terna, Simulation tools for social scientists: building agent-based models with swarm, *J. Artificial Societies and Social Simulation*. **1**, (1998).
57. S. Haykin, *Neural Networks: A Comprehensive Foundation (2nd ed.)*. (Prentice Hall, Englewood Cliffs, NJ, 1999).
58. G. Orr and K.-R. Mueller, Eds., *Neural Networks: Tricks of the Trade*. (Springer-Verlag, Berlin, Germany, 1998).
59. J. Principe. N. Euliano and W. Lefebre, *Neural and Adaptive Systems: Fundamentals Through Simulations*. (Wiley, New York, NY, 2000).
60. Y. Ishida, *Immunity-Based Computing: A Design Perspective*. (Springer-Verlag, Berlin, Germany, 2004).
61. P. Velleman and D. Hoaglin, *Applications, Basics, and Computing of Exploratory Data Analysis*. (Duxbury Press, Boston, MA, 1981).
62. V. Krebs, Mapping terrorist networks, *Connections*. **24**(3), 43–52, (2002).
63. J. Engene, Five decades of terrorism in europe: the tweed data set, *J. Peace Research*. **40**, 109–121, (2007).
64. M. Zhang and W. Li, Identifying potential synthesis cases in distributed excpert systems: a fuzzy logic approach, *Knowledge Based Systems*. **14**, 359–365, (2001).
65. F. Ren, M. Zhang and Q. Bai. A fuzzy-based approach for partner selection in multi-agent systems. In *Proc. 6th IEEE/ACIS Intl. Conf. on Computer and Information Science (ACIS'07)*, pp. 457–464, Los Alamitos, CA, (2007). IEEE Computer Society Press.

This page intentionally left blank